60459670

CONTROL DATA CORPORATION

NOS VERSION 2 REFERENCE SET

Volume 2

GUIDE TO SYSTEM USAGE

CDC® COMPUTER SYSTEMS:
CYBER 180
CYBER 170
CYBER 70
MODELS 71, 72, 73, 74
6000

COMMAND INDEX

APPEND 5-4; 7-7
ASCII 4-8; 8-5; 11-14
ATTACH 4-3; 5-7,14; 7-7
AUTO 2-17
BASIC 11-1
BEGIN 8-3; 9-25
BKSP 4-11
BYE 2-13
CATALOG 10-4
CATLIST 5-18
CHANGE 5-17
CHARGE 2-9
COBOL5 11-1
COMPASS 11-1
COPY 6-4
COPYBF 4-4
COPYBR 4-4
COPYCF 4-6
COPYCR 4-6
COPYEI 4-4
COPYSBF 4-7
DAYFILE 7-5
DEFINE 5-8,14; 7-7
DISPLAY 8-3,11
ELSE 8-3
ENDIF 8-2,3
ENDW 8-3,5
ENQUIRE 2-19,23; 7-7
EXECUTE 11-6
EXIT 7-1
FCOPY 3-8
FSE 12-3
FTN 11-1
FTNTS 2-4
FTN5 11-1
GET 4-2; 5-13; 7-7
GTR 10-7
IF 8-2,3
Job command 2-7
LABEL 6-2,6
LDSET 11-10
LIBEDIT 10-6,9
LIBGEN 10-4
LIBRARY 11-9
LIMITS 2-21
LINE 12-1
LISTLB 6-16
LOAD 11-6
MODIFY 13-2
Name call command 11-6; B-7
NEW 4-2
NOEXIT 7-2
NOGO 11-8
NORMAL 4-8
OLD 4-2; 7-7
ONEXIT 7-3
PERMIT 5-16
PURGE 5-10; 7-7
RENAME 4-13
REPLACE 5-2,6
RESOURC 6-9
RETURN 4-14
REVERT 8-3; 9-8
REWIND 4-12
ROUTE 3-7
SAVE 5-2,13
SCREEN 12-1
SCOPY 4-7
SET 8-3,10
SKIP 8-2,3
SKIPEI 4-11
SKIPF 4-10
SKIPFB 4-12
SKIPR 4-10
SLOAD 11-7
SUBMIT 2-14
UNLOAD 4-15; 6-10
UPROC 9-27
USER 2-8
VSN 6-12
WHILE 8-3,5

60459670

CONTROL DATA CORPORATION

NOS VERSION 2
REFERENCE SET

Volume 2

GUIDE TO SYSTEM USAGE

CDC® COMPUTER SYSTEMS:
CYBER 180
CYBER 170
CYBER 70
MODELS 71, 72, 73, 74
6000

REVISION RECORD

REVISION	DESCRIPTION
A (12-15-82)	Manual released. This manual reflects NOS Version 2.0 at PSR level 562.
B (10-14-83)	This manual reflects NOS Version 2.2 at PSR level 596/587. This revision adds an overview of the system explaining single-host and multihost arrangements and secured systems. Creating menu procedures is added. A new section introduces the use of Full Screen Editor and procedures in screen mode on the Viking 721 terminal. A new appendix, Accessing a Host, presents all aspects of login/logout. A new appendix, Debugging Aids, is added. Because revisions are extensive, no change bars or dots are used and all pages reflect the latest revision. This edition obsoletes all previous editions.
C (10-05-84)	This manual reflects NOS Version 2.3 at PSR level 617. New features described in this revision include expanded terminal support for the full-screen products and the alternate CATLIST display option. The section titled Execution Control Commands has been changed to Flow Control Commands. Sections 8 and 9 have been revised to help clarify the use of flow control commands and NOS procedures. A new section has been added on NOS libraries.

Publication No.
60459670

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED.

© 1982, 1983, 1984
by Control Data Corporation
All rights reserved
Printed in the United States of America

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Front Cover	-
Command Index	C
Title Page	-
2	C
3	C
4	C
5	C
6	B
7	B
8	B
9	C
10	C
11	C
12	C
13	C
14	C
1-1	C
1-2	C
1-3	C
1-4	C
2-1	C
2-2	C
2-3	C
2-4	C
2-5	C
2-6	C
2-7	C
2-8	C
2-9	C
2-10	C
2-11	C
2-12	C
2-13	C
2-14	C
2-15	C
2-16	C
2-17	C
2-18	C
2-19	C
2-20	C
2-21	C
2-22	C
2-23	C
3-1	B
3-2	B
3-3	C
3-4	B
3-5	B
3-6	B
3-7	C
3-8	B
3-9	C
4-1	C
4-2	B

PAGE	REV
4-3	B
4-4	C
4-5	C
4-6	B
4-7	B
4-8	B
4-9	C
4-10	C
4-11	B
4-12	C
4-13	B
4-14	B
4-15	C
5-1	B
5-2	B
5-3	B
5-4	B
5-5	B
5-6	B
5-7	B
5-8	B
5-9	B
5-10	B
5-11	B
5-12	B
5-13	C
5-14	C
5-15	C
5-16	C
5-17	C
5-18	C
5-19	C
5-20	C
5-21	C
5-22	C
6-1	C
6-2	C
6-3	B
6-4	C
6-5	C
6-6	C
6-7	C
6-8	C
6-9	C
6-10	C
6-11	C
6-12	C
6-13	C
6-14	C
6-15	C
6-16	B
6-17	B
7-1	C
7-2	C
7-3	C
7-4	C

PAGE	REV
7-5	C
7-6	C
7-7	C
7-8	C
8-1	C
8-2	C
8-3	C
8-4	C
8-5	C
8-6	C
8-7	C
8-8	C
8-9	C
8-10	C
8-11	C
8-12	C
8-13	C
8-14	C
8-15	C
9-1	C
9-2	C
9-3	C
9-4	C
9-5	C
9-6	C
9-7	C
9-8	C
9-9	C
9-10	C
9-11	C
9-12	C
9-13	C
9-14	C
9-15	C
9-16	C
9-17	C
9-18	C
9-19	C
9-20	C
9-21	C
9-22	C
9-23	C
9-24	C
9-25	C
9-26	C
9-27	C
10-1	C
10-2	C
10-3	C
10-4	C
10-5	C
10-6	C
10-7	C
10-8	C
10-9	C

PAGE	REV
10-10	C
10-11	C
11-1	C
11-2	C
11-3	C
11-4	C
11-5	C
11-6	C
11-7	C
11-8	C
11-9	C
11-10	C
11-11	C
11-12	C
11-13	C
12-1	C
12-2	C
12-3	C
12-4	C
12-5	C
12-6	C
12-7	C
12-8	C
12-9	C
12-10	C
12-11	C
12-12	C
12-13	C
12-14	C
12-15	C
12-16	C
12-17	C
12-18	C
12-19	C
12-20	C
12-21	C
12-22	C
12-23	C
12-24	C
12-25	C
12-26	C
12-27	C
12-28	C
13-1	C
13-2	C
13-3	C
13-4	C
13-5	C
13-6	C
13-7	C
13-8	C
13-9	C
13-10	C
13-11	C
13-12	C
13-13	C

PAGE	REV
13-14	C
13-15	C
13-16	C
13-17	C
13-18	C
13-19	C
13-20	C
13-21	C
13-22	C
13-23	C
13-24	C
13-25	C
13-26	C
13-27	C
A-1	B
A-2	B
A-3	B
A-4	C
A-5	B
A-6	B
A-7	B
A-8	B
A-9	B
A-10	B
A-11	B
A-12	C
A-13	B
A-14	B
A-15	B
B-1	B
B-2	C
B-3	B
B-4	B
B-5	B
B-6	B
B-7	B
B-8	B
B-9	C
B-10	B
B-11	B
B-12	B
C-1	B
C-2	C
C-3	C
C-4	C
C-5	C
C-6	C
C-7	C
C-8	C
C-9	B
C-10	B
C-11	B
C-12	B
C-13	B
C-14	B
C-15	B

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
C-16	B								
C-17	B								
C-18	C								
C-19	B								
C-20	C								
C-21	C								
C-22	B								
C-23	C								
C-24	C								
C-25	C								
C-26	B								
C-27	C								
C-28	C								
C-29	B								
C-30	B								
C-31	C								
C-32	B								
C-33	B								
C-34	C								
C-35	B								
C-36	C								
C-37	B								
C-38	C								
C-39	B								
C-40	B								
D-1	C								
D-2	B								
D-3	B								
D-4	B								
D-5	B								
D-6	C								
D-7	B								
D-8	B								
Index-1	C								
Index-2	C								
Index-3	C								
Index-4	C								
Index-5	C								
Index-6	C								
Index-7	C								
Comment Sheet	C								
Back Cover	-								

PREFACE

This guide describes the CONTROL DATA® Network Operating System (NOS) Version 2. NOS controls the operation of CDC® CYBER 180® Computer Systems, CDC CYBER 170® Computer Systems; CDC CYBER 70® Computer Systems, Models 71, 72, 73, and 74; and CDC 6000 Computer Systems.

ORGANIZATION AND AUDIENCE

This guide is the second volume of the NOS 2 Reference Set, which comprises four volumes. It is written for the applications or systems programmer who is not familiar with NOS. Volume 2 describes the general concepts of NOS and some of the utilities used with NOS. Section 1 introduces you to the system. Section 2 describes how you enter jobs into the system. Sections 3, 4, and 5 describe how you use files in processing jobs. You should read sections 1 through 5 before reading other sections within this guide. Sections 6 through 13 show additional features you can use when you are processing your jobs. These include information about using tapes, anticipating errors, creating procedures, modifying files, executing files, and using the Full Screen Editor.

Volume 1, Introduction to Interactive Usage, shows you how to enter, run, and correct programs and how to create, retrieve, and maintain permanent files at an interactive terminal. Other topics covered include physical terminal connection and procedures for entering and leaving the system. It is written for the person who has had little or no experience using computer systems in general and NOS in particular.

Volume 3, System Commands, describes the system commands that form the user interface with NOS. It is written for all NOS users.

Volume 4, Program Interface, describes the COMPASS program interface with NOS. Included are descriptions of function processors and macros available to COMPASS user programs. It is written for the experienced COMPASS applications or systems programmer.

The reader of each volume should have a knowledge of the material contained in the preceding volumes.

CONVENTIONS AND ASSUMPTIONS

In examples of actual terminal sessions shown in this guide, entries the user makes to the computer system are shown in blue. When you type characters at a terminal, you type them without pressing the shift key or any other key unless specifically instructed to do so. On most terminals, the characters you type will appear as lowercase letters. Therefore, in examples that show how a user's entries and the system's responses might look at a terminal, the user entries are shown in lowercase letters. The examples were produced on a standard ASCII display terminal. The vertical spacing in examples does not necessarily coincide with the spacing that appears on your terminal.

Interpret uppercase characters within command formats literally. Lowercase characters are variables.

When commands are shown in this guide, the formats usually do not include all the parameters available. We show only the more commonly used parameters. For the entire format, refer to Volume 3, System Commands.

The manuals of many NOS products use the term control statement; this guide uses the term command. You can consider the two synonymous.

Carriage return denotes the message transmission key on the keyboard. Depending on the terminal, this key may be the RETURN, CR, CARRIAGE RETURN, NEW LINE, SEND, NEXT, or ETX key. The system responds to a carriage return by positioning the cursor to the first character position on the next line.

The notation CTRL/x directs you to press the control key (which is labeled CTRL, CNTL, CNTRL, or similar characters) on the terminal and, while holding this key down, to press the key specified by x. For example, CTRL/T means press and hold the control key while you press the T key.

RELATED PUBLICATIONS

Following is a list of NOS manuals that you may be interested in.

The NOS Version 2 Manual Abstracts is a pocket-sized manual containing brief descriptions of the contents and the intended audience of all NOS 2 manuals and NOS 2 optional products manuals. The abstracts can be useful in determining which manuals are appropriate for your use.

Control Data publishes a Software Publications Release History of all software manuals and revision packets it has issued. This history lists the revision level of a particular manual that corresponds to the level of software installed at the site.

Control Data manuals are available through Control Data sales offices or Control Data Literature Distribution Services, 308 North Dale Street, St. Paul, MN 55103.

Control Data Publication	Publication Number
APL Version 2 Reference Manual	60454000
BASIC Version 3 Reference Manual	19983900
COBOL Version 5 Reference Manual	60497100
COMPASS Version 3 Reference Manual	60492600
CYBER Interactive Debug Version 1 Reference Manual	60481400
CYBER Loader Version 1 Reference Manual	60429800
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aids Program Version 1 Reference Manual	60483000
FORTRAN Version 5 Reference Manual	60481300
Modify Version 1 Instant Manual	60450200
Modify Version 1 Reference Manual	60450100
NOS Version 2 Application Installation Handbook	84002760
NOS Version 2 Applications Programmer's Instant	60459360
NOS Version 2 Diagnostic Index	60459390
NOS Version 2 Full Screen Editor User's Guide	60460420
NOS Version 2 Installation Handbook	60459320
NOS Version 2 Manual Abstracts	60485500

Control Data Publication	Publication Number
NOS Version 2 Network Terminal User's Instant	60459380
NOS Version 2 Operator/Analyst Handbook	60459310
NOS Version 2 Reference Set, Volume 1, Introduction to Interactive Usage	60459660
NOS Version 2 Reference Set, Volume 3, System Commands	60459680
NOS Version 2 Reference Set, Volume 4, Program Interface	60459690
NOS Version 2 Screen Formatting Reference Manual	60460430
NOS Version 2 System Maintenance Reference Manual	60459300
NOS Version 2 System Overview	60459270
NOS Version 2 Systems Programmer's Instant	60459370
PL/I Version 1 Reference Manual	60388100
Remote Batch Facility Version 1 Reference Manual	60499600
Software Publications Release History	60481000
Text Editor Version 1 Reference Manual	60436100
Update Version 3 Reference Manual	64499000

DISCLAIMER

This guide describes a subset of the features and parameters documented in Volume 3, System Commands, the various utility reference manuals, and the high-level programming language reference manuals. Control Data cannot be responsible for the proper functioning of any features or parameters not described in these manuals.

CONTENTS

1. OVERVIEW OF THE SYSTEM 1-1

Software Features 1-1
Hardware Configurations 1-2
    Single-Host Arrangements 1-2
    Multihost Arrangement 1-3
Secured System 1-4

2. JOB PROCESSING 2-1

Batch Jobs 2-5
Interactive Jobs 2-5
Job Structure 2-7
    Entering the System 2-7
        Batch Jobs 2-7
        Interactive Jobs 2-10
    Completing a Task 2-11
        Batch Jobs 2-12
        Interactive Jobs 2-13
    Leaving the System 2-13
        Batch Jobs 2-13
        Interactive Jobs 2-13
Submit Files 2-14
    Structuring a Submit File 2-15
    Creating a Submit File 2-17
    Monitoring a Submit File 2-19
Using System Resources 2-21

3. FILES 3-1

File Names 3-2
Logical File Structure 3-2
    Records 3-3
    Structuring a File 3-3
        Beginning-of-Information (BOI) 3-3
        End-of-Record (EOR) 3-4
        End-of-File (EOF) 3-4
        End-of-Information (EOI) 3-4
Files INPUT and OUTPUT 3-5
    File INPUT 3-5
        Interactive File INPUT 3-5
        Batch File INPUT 3-6
    File OUTPUT 3-6
    Routing Your Files 3-7
Temporary and Permanent Files 3-9

4. LOCAL FILES 4-1

Creating Local Files 4-1
    NEW Command 4-2
    OLD Command 4-2
    GET Command 4-2
    ATTACH Command 4-3
    Copying Commands 4-4
        Copying Binary Information 4-4
        Copying Coded Information 4-6
        Copying Magnetic Tapes 4-7
Using Uppercase and Lowercase Characters 4-8
Positioning Local Files 4-8
    SKIPR Command 4-10
    SKIPF Command 4-10
    SKIPEI Command 4-11
    BKSP Command 4-11
    SKIPFB Command 4-12
    REWIND Command 4-12
Changing Names of Local Files 4-13
Releasing Local Files 4-13
    RETURN Command 4-14
    UNLOAD Command 4-15

5. PERMANENT FILES 5-1

Creating an Indirect Access File 5-2
Accessing an Indirect Access File 5-3
Adding Information to an Indirect Access File 5-4
Modifying an Indirect Access File 5-6
Creating a Direct Access File 5-8
Accessing a Direct Access File 5-9
Modifying a Direct Access File 5-9
Manipulating Files 5-10
Removing Permanent Files from the System 5-10
Granting Other Users Access to Files 5-12
    Indirect Access Files 5-13
    Direct Access Files 5-14
    Specific Users 5-16
Changing Permanent File Characteristics 5-17

Obtaining Permanent File Information 5-18
Listing Permanent File Names 5-18
Listing General Permanent File Information 5-19
Listing Accesses by Other Users 5-22

6. MAGNETIC TAPE PROCESSING 6-1

Reading a Labeled Tape 6-2
Writing on a Labeled Tape 6-6
Copying from One Tape to Another Tape 6-9
Multivolume Files 6-12
Multifile Sets 6-14

7. ERROR CONTROL 7-1

Job's Dayfile 7-4
Error Control by Individual Commands 7-7
Errors Detected by the Hardware 7-8

8. FLOW CONTROL COMMANDS 8-1

Skipping Commands 8-1
SKIP and ENDIF Commands 8-2
IF and ENDIF Commands 8-2
IF, ELSE, and ENDIF Commands 8-3
Looping Commands 8-5
WHILE and ENDW Commands 8-5
NOS Procedure Commands 8-6
Using Expressions in Flow Control Commands 8-8
SET Command 8-10
DISPLAY Command 8-11
Functions 8-12
File Function 8-12
Number Function 8-15

9. NOS PROCEDURES 9-1

Why Use Procedures? 9-1
Procedure Header Directive and REVERT Command 9-2
Parameter Substitution 9-3
Special Substitution Characters 9-5
Inhibit Character (#) 9-5
Concatenation Character (_) 9-6
Literal Delimiter ($) 9-7
Terminating Procedures 9-8
REVERT Command 9-8
Error Control 9-9
Interactive vs. Noninteractive Procedures 9-10
Interactive Procedures 9-10
Interactive Procedure Header Directive 9-12
Checklist Patterns 9-15
Procedure and Parameter Help 9-19
Menu Procedures 9-19
Using Menu Procedures 9-20
Menu Procedure Header Directive 9-22
Using Procedures 9-22
Multi-Record Procedure Files 9-25
BEGIN Command 9-25
Prologue Procedures 9-27

10. LIBRARIES 10-1

Types of NOS Libraries 10-1
Library Sets 10-2
Creating and Maintaining User Libraries 10-3
Calling LIBGEN 10-4
Creating a New User Library 10-4
Creating an Updated User Library 10-6
Editing a File Using LIBEDIT 10-7

11. FILE EXECUTION 11-1

Program Commands 11-1
Loading Programs 11-2
Types of Loading 11-3
Loading Processes 11-4
Loading Sequences 11-5
Basic Loading 11-6
Name Call Loading 11-6
Loading Programs from Multiple Files 11-6
Terminating a Loading Sequence 11-8
Entering Loading Sequences from the Terminal 11-9
Loading Programs from Libraries 11-9
Library Search Order 11-11
Additional Loader Options 11-13

12. SCREEN MODE 12-1

Using the Full Screen Editor in Screen Mode 12-2
Starting FSE 12-3
Creating Files 12-3
Entering Directives 12-5
Editing Files 12-7
Inserting and Deleting 12-8
Locating Text 12-8
Replacing Text 12-9
Recovering from Mistakes 12-9
Copying Lines 12-10
Getting Online Help 12-12
Using Procedures in Screen Mode 12-14
Using an Interactive Procedure 12-15
Getting Help in an Interactive Procedure 12-20
Formatting the Screen with Interactive Procedures 12-23
Using a Menu Procedure 12-24
Getting Help in a Menu Procedure 12-26
Multiple-Paged Screens 12-28

13. SOURCE FILE MAINTENANCE 13-1

FSE 13-1
Source Code Maintenance Utilities 13-4
Modify Concepts 13-5
Prepared Source Files 13-5
Program Libraries 13-5
Directives 13-6
Output Files 13-7
General Modify Job Structure 13-8
Using Modify 13-10
Preparing Your Source File 13-10
Calling Modify 13-12
Creating a Program Library 13-12
Generating a Compile File 13-14
Making Changes to the Program Library 13-16
Adding and Deleting Lines 13-20
Deleting Effects of Modification Sets 13-26

APPENDIXES

A. CHARACTER SETS A-1

Character Set Anomalies A-2
Character Set Tables A-2
Interactive Jobs A-3
Batch Jobs A-3
Jobs Using Line Printers A-4
Jobs Using Magnetic Tape A-12

B. GLOSSARY B-1

C. ACCESSING A HOST C-1

Brief Description of Access C-2
Step 1 - Gathering Information C-2
Step 2 - Setting Up Your Terminal C-2
Step 3 - Connecting Your Terminal to the Network C-3
Step 4 - Identifying Your Terminal to the Network C-5
Step 5 - Selecting a Host C-5
Step 6 - Connecting to the Selected Host C-6
Step 7 - Identifying Yourself and Your Terminal to the Host C-6
Step 8 - Selecting a Network Application C-6

Detailed Description of Access C-7
Step 1 - Gathering Information C-7
Step 2 - Setting Up Your Terminal C-12
Step 3 - Connecting Your Terminal to the Network C-13
Step 4 - Identifying Your Terminal to the Network C-14
Procedure for Asynchronous Terminals C-14
Procedure for HASP Terminals C-15
Procedure for Mode 4 Terminals C-17
Procedure for Bisynchronous Terminals C-17
Step 5 - Selecting a Host C-18
Selecting a Host C-18
Selecting a Specific Host Path C-19
Controlling the Display of Host Paths C-20
Step 6 - Connecting to the Selected Host C-22
Step 7 - Identifying Yourself and Your Terminal to the Host C-22
Automatic Login C-24
Login Dialog C-24
Step 8 - Selecting a Network Application C-27
Automatic Connection C-27
Manual Connection C-28

Abbreviated Login and Application Connection C-30
Switching to a Different Application C-31
Direct Switches C-31
Indirect Switches C-31
Dialog for Direct and Indirect Switches C-32
Restarting Login Identification C-32
Direct Restarts C-32
Indirect Restarts C-33
Secure Login Restarts C-33
Disconnecting from a Host C-34
Disconnecting with Application Commands C-34
Disconnecting with Host Commands C-35
Disconnecting with a Network Software Command C-36
Reconnecting to a Host or Connecting to a Different Host C-36
Disconnecting from One Host and Connecting to a Different Host C-36
Interruptions C-37
Communicating with the Network Operator C-37
Suspensions of Communication C-38
Communication Failures C-39
Application Failures C-39
Disconnection from an Application and/or Host C-39
Communication Failures C-40

D. DEBUGGING AIDS D-1

Exchange Package Dumps D-1
Using Dumps D-5

INDEX

FIGURES

1-1 Single-Host Arrangement 1-2
1-2 Single-Host Arrangement with Link to a Remote Host 1-3
1-3 Multihost Arrangement 1-3
2-1 Job Processing 2-3
2-2 Structuring Input 2-12
2-3 System Reformatting of a Submit File 2-16
2-4 Creating a Submit File 2-17
2-5 Correcting Mistakes in Submit Files 2-18
2-6 Determining the Status of a Submit File 2-19
2-7 Determining the Status of More Than One Submit File 2-20
2-8 Resource Allocations and Limits 2-22
2-9 Time Limit 2-23
3-1 Example of a Multifile File 3-1
3-2 EOR, EOF, and EOI Representations on Punched Cards 3-4
3-3 File INPUT for Batch Jobs 3-6
3-4 Printing an Uppercase and Lowercase File 3-8
4-1 The NEW, OLD, GET, and ATTACH Commands 4-3
4-2 File Creation with Binary Copying Commands 4-5
4-3 Copying Coded Records 4-9
4-4 SKIPR Command 4-10
4-5 BKSP Command 4-11
4-6 Releasing Local Files 4-14
4-7 Releasing Local Files Using Special Parameters 4-15
5-1 Accessing and Creating Indirect Access Files 5-3
5-2 Modifying an Indirect Access File 5-7
5-3 Manipulating Local and Permanent Files 5-11
5-4 Listing Your File Names 5-19
5-5 Full CATLIST Example 5-21
5-6 Listing Accesses by Other Users to a Private File 5-22
6-1 Copying a Tape File to Disk 6-5
6-2 Copying a File to Tape 6-7
6-3 Adding a Record to a Tape File 6-8
6-4 Copying from One Tape to Another Tape 6-10
6-5 Tape Copying with Limited Resources 6-11
6-6 Writing a Multivolume File 6-13
6-7 Listing Labels 6-17
7-1 Batch Job Error Control Commands 7-1
7-2 Exit Processing 7-2

7-3 Example of NOEXIT Command 7-2
7-4 Dayfile from a Batch Job 7-4
7-5 Interactive Dayfile Example 7-5
7-6 Using the DAYFILE Command in a Submitted Job 7-6
8-1 IF/ENDIF Processing 8-4
8-2 IF/ELSE/ENDIF Processing 8-4
8-3 WHILE/ENDW Processing 8-6
8-4 Using the SET Command with the WHILE and ENDW Commands 8-7
8-5 Using the SET and DISPLAY Commands 8-13
8-6 Use of the FILE Function 8-14
8-7 Use of the NUMBER Function 8-15
9-1 Menu Procedure Display 9-11
9-2 Interactive Procedure File Format 9-13
9-3 Procedure LEARN Examples 9-14
9-4 Checklist Patterns 9-16
9-5 Checklist Pattern Usage 9-18
9-6 Procedure DISPOSE 9-21
9-7 Procedure DISPOSE (with Help) 9-21
9-8 DISPOSE Dayfile Listing 9-22
9-9 Getting Procedure Help 9-23
9-10 Files MYPRINT, MYPUNCH, and MYPLOT 9-24
9-11 Multi-Record Procedure File 9-26
10-1 Using LIBGEN 10-3
10-2 Creating a New User Library 10-5
10-3 Adding Records to a User Library 10-6
10-4 Source Program File NEWSRC 10-10
10-5 The LIBEDIT Directives 10-10
10-6 Updating a User Library 10-11
11-1 Loader Input and Output 11-2
11-2 Library Search Order 11-12
12-1 Viking 721 Terminal Keyboard 12-2
12-2 FSE Screen 12-3
12-3 File BIKE 12-6
12-4 File to Be Edited 12-7
12-5 Copying Lines 12-11
12-6 HELP File for Directive LOCATE 12-13
12-7 Example of an Interactive Procedure 12-15
12-8 Screen Display of Procedure Prompts 12-16
12-9 Screen Display of Entering Parameters 12-18
12-10 Screen Showing an Incorrect Parameter Entry 12-19
12-11 Screen Displaying Parameter HELP Text 12-20
12-12 Screen Displaying HELP Text for the Procedure Description 12-21
12-13 Screen Displaying Parameter HELP Text 12-22
12-14 Example of a Menu Procedure 12-24
12-15 Screen Displaying Menu Options 12-25
12-16 Screen Displaying Menu HELP Text 12-26
12-17 Screen Displaying HELP Text for a Menu Option 12-27
12-18 Display of Function Keys for Procedures 12-28
13-1 Using FSE 13-2
13-2 General Structures for Batch and Interactive Jobs 13-8
13-3 Example of Deck Names and EORs 13-10
13-4 Example of a Common Deck 13-11
13-5 Creating a Program Library 13-13
13-6 Generating a Compile File 13-15
13-7 Using the EDIT Directive 13-17
13-8 Using the IDENT and DECK Directives 13-19
13-9 Locating Lines within the Compile File 13-20
13-10 Locating Lines within a Modified Compile File 13-21
13-11 Using the INSERT Directive 13-23
13-12 Using the DELETE Directive 13-25
13-13 Using the YANK Directive 13-27
C-1 Separate Acoustic Coupler C-3
C-2 Data Set Built into the Terminal C-3
C-3 Data Sets with Switches and Buttons C-4
C-4 Terminal with Built-in Acoustic Coupler C-4
C-5 Interactive Display Console C-8
C-6 Interactive Printer Console C-9
C-7 Remote Batch Terminal C-9
C-8 Host Availability Display C-20
C-9 Sample Login from an Asynchronous Terminal C-23
D-1 Exchange Package Dump D-2
D-2 Exchange Package Dump for Model 176 D-2
D-3 Program Listing and Symbolic Reference Map D-6
D-4 Partial Load Map D-7
D-5 Dayfile D-7
D-6 Exchange Package Dump D-8
D-7 Central Memory Dump D-8

TABLES

8-1 Flow Control Commands 8-3
8-2 Operators in NOS Expressions 8-8
8-3 Some Commonly Used Symbolic Names 8-9
10-1 Maximum Size of Global Library Set 10-2
10-2 LIBEDIT Directives 10-8
11-1 BASIC, FORTRAN, and COMPASS Commands 11-1
A-1 Character Sets for Interactive Jobs A-5
A-2 Character Sets for Batch Jobs A-7
A-3 ASCII to 6/12-Bit Display Code Conversion A-10
A-4 Nine-Track ASCII Coded Tape Conversion A-13
A-5 Nine-Track EBCDIC Coded Tape Conversion A-14
A-6 Seven-Track Coded Tape Conversions A-15

SYSTEM OVERVIEW 1

The purpose of this manual is to teach you how to use the Network Operating System (NOS) to perform job processing tasks. The first five sections are primarily concerned with explaining the basic NOS processing concepts of system commands, files, and jobs. If you are not already familiar with the NOS system, you should read these five sections before you read any of the later sections.

The remainder of the manual provides you with basic information on some of the more frequently used system utilities and features. For many users, the discussions of these utilities and features will provide sufficient information to perform most of their daily processing tasks. If you need more information on any of these subjects, we provide references for further reading.

SOFTWARE FEATURES

The major utilities and features described in this manual are:

NOS Procedures

NOS procedures are similar to program subroutines. A procedure is a sequence of NOS commands that can be called and executed in the same way that a program calls a subroutine. Variable values such as file names or processing options can be passed to a procedure by way of procedure parameters. NOS procedures provide a simple way of performing repetitive system processing operations.

Libraries

Libraries are a convenient way of storing programs, routines, NOS procedures, data files, and text files. This manual describes the LIBGEN and LIBEDIT utilities used to create and maintain libraries.

CYBER Loader

Before you can execute a program on NOS, you must ensure that all required program segments and routines are available to be loaded into central memory. Except for very simple programs, this usually requires the use of the CYBER Loader utility. In this manual we describe how to perform basic loader operations of loading program data from files and libraries.

Tape Processing

The tape processing section describes the basics of storing and retrieving files from magnetic tape.

Screen Mode Display

The screen mode feature makes it possible to use full-screen data displays for collecting, displaying, or modifying various types of data. NOS provides three screen mode features, the Full Screen Editor (FSE), full-screen display of NOS procedures, and the Screen Formatting feature. The Screen Formatting feature provides full-screen input and output capabilities for application programs.

MODIFY Utility

The MODIFY utility simplifies maintenance of lengthy source code libraries. With its use of common decks, global modification operations, and a record of previous modifications, MODIFY significantly reduces the time and effort required to maintain large source code files.

HARDWARE CONFIGURATIONS

This section gives you a general description of the components of your computer system. The basic components of a system are terminals, a network, and one or more hosts. Each of these components is a combination of hardware and software. You usually communicate with the host by entering information at a terminal, which may be located away from the host. The network serves as a connection between your terminal and the host.

The host consists of a mainframe, the Network Operating System (the software that serves as the mechanism for getting jobs done), and applications (software that performs a variety of specialized tasks). You may have access to one or more hosts, and each host may have one or more host nodes. A host node is the path by which you are connected to the host. The host and host node may be selected by default, or you can request one of the hosts or host nodes listed on a display that may show up before login.

SINGLE-HOST ARRANGEMENTS

Figure 1-1 shows a typical single-host arrangement. The host may have one or more host nodes; in this figure, two host nodes are shown.

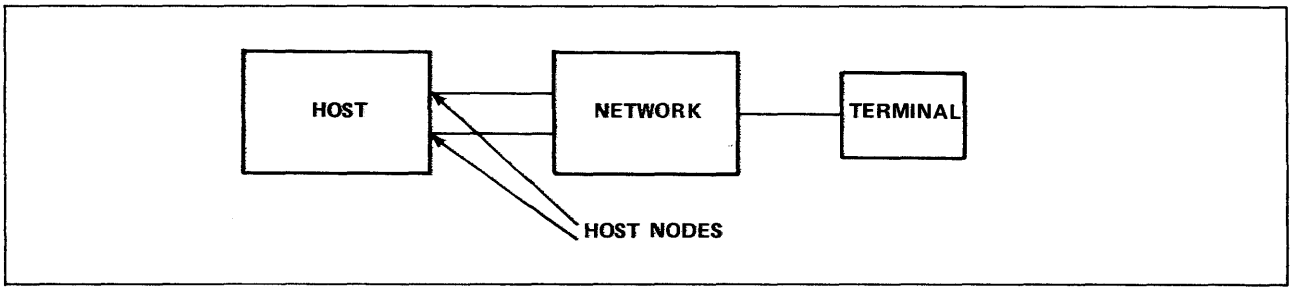


Figure 1-1. Single-Host Arrangement

Figure 1-2 is an example of a single-host arrangement in which you have access to a linked second host. You can log in only to your host. Using special system commands, however, you can transfer files back and forth across the link and process jobs on the linked host, as well as on the host you are logged in to.

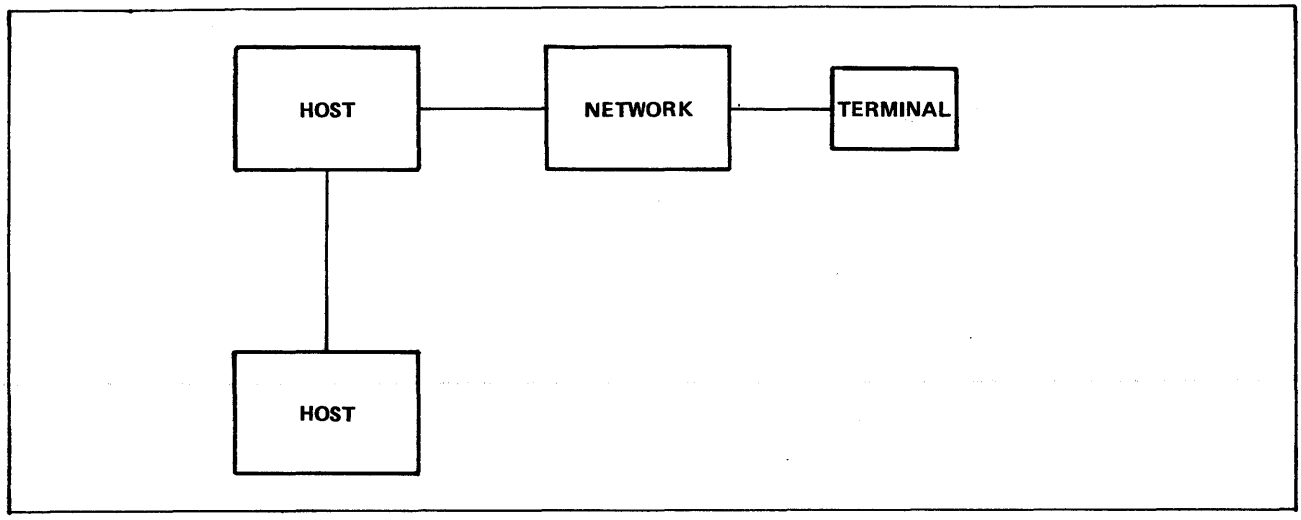


Figure 1-2. Single-Host Arrangement with Link to a Remote Host

MULTIHOST ARRANGEMENT

Figure 1-3 shows an example of a multihost arrangement. This differs from a single-host with a linked remote host in that you can log in directly to different hosts. If you have a multihost arrangement, site personnel determine whether you are automatically connected to a host or must select a host at login time.

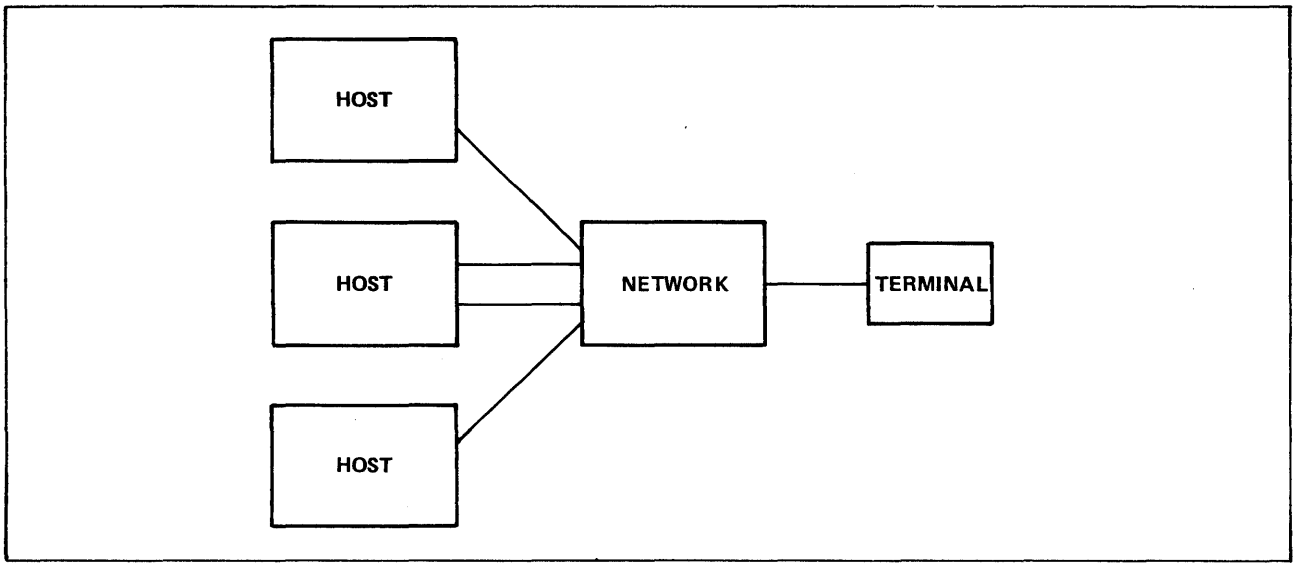


Figure 1-3. Multihost Arrangement

SECURED SYSTEM

Your site may have installed a secured system. In a secured system, NOS enforces access controls based on security access levels and categories. The system automatically checks clearance levels before allowing access to information.

Other constraints on user jobs and files depend on the type of peripheral equipment, the type of terminals, and the type of job processing. There may also be differences in your terminal login procedure (refer to appendix C). Your site should provide guidelines on the use of security levels and categories, login procedures, and security constraints enforced.

This manual, unless stated otherwise, assumes you are logged into a single-host, unsecured computer system. If you are using this guide for a secured system, assume you are using a default access level and therefore do not need to specify security parameters on file management commands. If you want to access or create information at other access levels, refer to Volume 3, System Commands, for information on security parameters.

JOB PROCESSING 2

In this section we describe how you communicate with the Network Operating System (NOS).

Your purpose in using a computer system is to accomplish certain tasks. These tasks often include editing a program file; editing a text file; assembling, compiling, or executing a program; or any of a number of other tasks.

In the NOS operating system, you accomplish your tasks by communicating with the system through a set of system commands. It is by entering a series of commands that you tell the system what you want to do. An example of a system command is:

```
GET,AFILE.
```

This command instructs the system to find a file called AFILE in permanent storage and make a copy of the file available to your job. Once it is available to your job, you can use other commands to modify AFILE, execute it, replace the permanent file copy, and so on. The primary purpose of this guide is to teach you how to use system commands.

System commands must be entered to the system from a job. A job is essentially an accounting concept that the system uses to keep track of system resources. Whenever you initiate a job by submitting a card deck or logging in to an interactive terminal, you must first identify yourself to the system. You do this by specifying your user name and password. Once you have identified yourself to the system, the system can check your authorized security level, keep track of system resources that you use, identify any files you may have saved in permanent storage, and so on.

There are two types of jobs:

- Batch jobs.
- Interactive jobs.

In a batch job, you enter all commands at one time as a unit. You do this either by using a punched card deck or by submitting a file containing a job from an interactive terminal. If you use punched cards, you either enter the card deck yourself at a remote batch card reader or give the cards to site personnel, who will enter your job at the central site. A batch job that you submit for execution from an interactive terminal executes separately from the interactive job that you are running when you submit the job. Submitting jobs is discussed later in this section.

In an interactive job, you communicate with the system by entering commands at an interactive terminal. The system generally processes each command right after you enter it, and you can determine if it was processed successfully before you enter the next command.

Most system commands involve the manipulation of one or more files. The file is the basic information storage unit for NOS users. Each file has a name by which it is identified. Information typically stored in files includes:

- Programs
- Text information
- NOS procedures
- Databases
- Batch jobs
- Job output

If you do not already know how to create a file using the Full Screen Editor (FSE), you may want to take some time now to become familiar with FSE. You will probably want to create some files and try out many of the NOS commands we present to you.

There are many other ways of creating files besides FSE. We will show you some of these a little later.

Figure 2-1 contains an example of a batch job and an example of an interactive job; the two jobs compile and execute similar FORTRAN programs (we describe the commands later). Compare the two jobs; although accessing the system is different, batch and interactive job processing are very similar.

BATCH JOBS

Card Deck

```
EXAMPLE.
USER,BMR2301,RRRR.
GET,EXAMPLE.
FTN5,I=EXAMPLE,L=0.
LGO.



7/8/9

8

6/7/8/9
```

Description

The first three cards identify the job and the user.

The next three cards contain commands that get, compile, and execute the FORTRAN program called EXAMPLE. They are in the order in which they are to be processed.

A card that has the numbers 7, 8, and 9 punched in column 1 indicates an end-of-record (EOR). An EOR separates the preceding commands from other records in the job file.

A card with the number 8 punched in column 1 is the input for the FORTRAN program.

A card that has the numbers 6, 7, 8, and 9 punched in column 1 indicates an end-of-file (EOF). An EOF marks the end of the job file.

Output from Batch Job

```
 LIST OF ODD NUMBERS

1
3
5
7
9
11
13
15

12.59.37.EXAMPLE.
12.59.37.UCCR, 06, 024,
12.59.37.USER,BMR2301,.
12.59.37.GET,EXAMPLE.
12.59.39.FTN5,I=EXAMPLE,L=0.
12.59.39.     56000 CM STORAGE USED.
12.59.39.     0.021 CP SECONDS COMPILATION TIME.
12.59.39.LGO.
12.59.41.    STOP
12.59.41.     17700 MAXIMUM EXECUTION FL.
12.59.41.      .013 CP SECONDS EXECUTION TIME.
12.59.41.UEAD,      0.002KUNS.
12.59.41.UEPF,      0.009KUNS.
12.59.41.UEMS,      1.504KUNS.
12.59.41.UECP,      0.113SECS.
12.59.41.AESR,      2.409UNTS.
12.59.41.$OUT(*/OP=E)
12.59.41. NO FILES PROCESSED.
12.59.41.$DAYFILE(OUTPUT,JT=D)
13.00.51.UCLP, 42, 071,      0.128KLNS.
```

Description

The system prints the output at the central site. The FORTRAN program uses an input of 8, which causes eight odd numbers to be printed.

Part of the output is a list of the system's activity called the dayfile. Section 7 gives you more information on the dayfile.

Figure 2-1. Job Processing (Sheet 1 of 2)

INTERACTIVE JOB

Terminal Session†

```
WELCOME TO THE NOS SOFTWARE SYSTEM.
COPYRIGHT CONTROL DATA 1978, 1984.
84/07/07. 15.59.40. T01A63
CDC NETWORK OPERATING SYSTEM.          NOS 2
FAMILY:   ,bmr2301,rrrr,iaf
JSN: ACMB, NAMIAF

 READY.
batch
RFL,0.
/get,example
/ftn5,i=example,l=0
     0.022 CP SECONDS COMPILATION TIME.
/lgo
   LIST OF ODD NUMBERS
  HOW MANY NUMBERS
? 8
 1
 3
 5
 7
 9
 11
 13
 15

      .009 CP SECONDS EXECUTION TIME.
/dayfile
 14.31.16.AJAI.
 14.31.17.USER,BMR2301,,NOSHOP.
 14.31.17.RECOVER,OP=T.
 14.31.33.RFL,0.
 14.31.56.GET,EXAMPLE.
 14.32.29.FTN5,I=EXAMPLE,L=0.
 14.32.29.     56000 CM STORAGE USED.
 14.32.29.     0.022 CP SECONDS COMPILATION TIME.
 14.32.45.LGO.
 14.33.43.    STOP
 14.33.43.     16100 MAXIMUM EXECUTION FL.
 14.33.43.     0.009 CP SECONDS EXECUTION TIME.
 14.34.08.DAYFILE.
 USER DAYFILE PROCESSED.
/bye
UN=BMR2301  LOG OFF  13.25.46.
JSN=ACMB     SRU-S       2.511
CHARACTERS=    1.258KCHS
IAF     CONNECT TIME 00.03.01.
LOGGED OUT.

HOST DISCONNECTED    CONTROL CHARACTER=(ESC)
NPU NODE= :1 TERMINAL NAME= T01A69
HOST     NODE  SELECTED/   STATUS
               CONNECTED
ARHNOS3   7              NOT AVAILABLE
ARHNOS    1      S       AVAILABLE
ARNOS     2              NOT AVAILABLE
ENTER INPUT TO CONNECT TO HOST
```

Description

The user logs in. The data entered is similar to the first three cards in a batch job.

The user enters commands to get, compile, and execute the FORTRAN program EXAMPLE. The system asks for input (the question mark means the system is waiting for the user to enter information). The user enters the digit 8. Then, the system prints eight odd numbers.

The user enters the DAYFILE command to list the system's activity. The list is similar to the dayfile that is printed at the end of a batch job.

The user logs out; the job is finished.

†For all interactive terminal sessions in this guide, user input is in blue.

Figure 2-1. Job Processing (Sheet 2 of 2)

BATCH JOBS

A batch job entered using a card deck can be either local or remote.

A local batch job is a job you submit as a card deck to be read at the central site. The system usually prints the output on a central site printer. We describe the structure of a local batch job later.

A remote batch job is a job you submit from a remote batch terminal (a special type of terminal usually located away from the central site). The output for remote batch jobs is usually returned to the terminal from which you entered the job; however, you can choose to have it printed at the central site. The structure of the card deck you enter for a remote batch job is similar to the structure of the card deck used for a local batch job. We do not give detailed information on remote batch job processing in this guide; refer to the Remote Batch Facility Reference Manual for more information.

A job you submit from another job is known as a deferred batch job. You can either retrieve the output from the submitted job at the terminal, or you can have it printed at the central site. The structure of the file you submit is discussed later in this section under Submit Files.

INTERACTIVE JOBS

Generally everything you do on a terminal from the time you log in to the system until you log out is a job. When you log in, the system assigns you to a subsystem chosen by site personnel. Each subsystem allows you to do particular things and to use some of the system commands (all system commands are described in Volume 3, System Commands). You can change to a different subsystem at any time during your session by entering the subsystem name. The subsystems are:

Name	Description
Access	Enables you to communicate with other terminals. You must be specifically validated to use this subsystem.
BASIC	Enables you to easily create, update, and execute BASIC programs. For more information on the BASIC subsystem, refer to Volume 1, Introduction to Interactive Usage.
Batch	Enables you to use the system commands, the commands that call compilers, and the commands that call system utilities.
Execute	Enables you to execute a previously compiled program.
FORTRAN	Enables you to easily create, update, and execute FORTRAN 5 programs that are prefaced by line numbers. The FORTRAN subsystem is designed for the beginning programmer. A program that runs under the FORTRAN subsystem can also be run under the batch subsystem using the FTN5 command (including the SEQ parameter). The FTN5 command provides you with more options than does the FORTRAN subsystem. In this guide, we use the FTN5 command under the batch subsystem, rather than the FORTRAN subsystem. For more information on the FORTRAN subsystem, refer to Volume 1, Introduction to Interactive Usage.

Name	Description
FTNTS	Enables you to easily create, update, and execute FORTRAN 4 programs that are prefaced by line numbers. The FTNTS subsystem is designed for the beginning programmer. A program that runs under the FTNTS subsystem can also be run under the batch subsystem using the FTN command (including the SEQ parameter). The FTN command provides you with more options than does the FTNTS subsystem.
Null	Enables you to use commands and perform file manipulations without subsystem association.

Subsystems are only available in an interactive job. When you are in the batch subsystem, you have access to all the capabilities of NOS. You can perform such activities as positioning files, copying from one file to another, and compiling programs from a terminal.

To enter the batch subsystem, you log in and then type:

```
BATCH
```

The system responds:

```
RFL,0.
/
```

The RFL,0 indicates that the system determines the field length for each entry (field length is the amount of central memory that is used for processing entries). In most cases, you do not need to specify the field length.

The slant character signifies that the system is ready for a command. If the system takes some time to process your command, you can enter a second command, or a series of commands, before the system prompts you with a slant. If you do, the system displays a slant when it completes execution of the first command; then it begins execution of the second command, displays a slant when it completes execution of the second command, and so on.

In the batch subsystem, no command can exceed 80 characters, and you must enter the entire command name even though it may have an abbreviated form.

In this guide, the terminal sessions that do not show the login assume that you are in the batch subsystem.

If you are in a subsystem other than batch, and you want to enter commands that are usually allowed only under the batch subsystem, you can precede the command by:

```
X,
```

This forces the system to accept the command. For example, entering the command:

```
CATALOG,LOOK
```

while you are in the null subsystem, produces the response:

```
INCORRECT COMMAND
```

However if you enter:

```
X,CATALOG,LOOK
```

the system processes the command (CATALOG gives you information about file LOOK).

JOB STRUCTURE

Every job, whether it is batch or interactive, must:

- Enter the system.
- Complete its task.
- Leave the system.

ENTERING THE SYSTEM

The information gathered by the system at the beginning of both batch and interactive jobs serves the purpose of identifying you and your job.

Batch Jobs

In a batch job, the system allows you access only if the first three cards in the deck contain no errors.

The first card, the Job command, specifies the job name. Its basic format is:

```
ujn.
```

ujn is a user job name that you choose. It can be from one to seven alphanumeric characters in length and must begin with a letter. This name appears on the banner page of your output after UJN = . The following examples are valid Job commands.

```
MYJOB.
A.
JONES.
JOB2323.
```

Optional parameters for the Job command specify the special needs of the job. They are not necessary for the beginning user.

The second card in the deck is the USER command. It establishes:

- That you are allowed to use the system.
- What system resources you can use (we discuss the system resources later).
- The location of your permanent files.

The USER command's format is:

```
USER,username,password,familyname.
```

Parameter	Description
username	The user name assigned to you by site personnel.
password	The password associated with your user name.† No one can access the system using your user name, unless they also know your password. The password is alphanumeric, its maximum length is seven characters, and its minimum length is four characters (site personnel may change the minimum value). Originally, your password is assigned by site personnel. If site personnel authorize it, you can change your password (we discuss authorization later). For more information on how to change your password, refer to the PASSWOR command in Volume 3, System Commands.
familyname	The name of the family of permanent file devices on which your permanent files reside. Site personnel determine the name you enter as the family name. Many sites have only one family. This means that all the users' permanent files reside on the same group of devices. If your site has more than one family, your files will usually reside on the default family. This means you can omit this parameter, unless you know that your site has more than one family and that your permanent files are on a device other than a default family device. In this guide, we use the default family for all examples.

The USER command parameters are order-dependent (that is, they must appear in the order shown in the command format).

Example:

If you are assigned the user name VHN35, if your password is VERN, and if your permanent files are on the default family, your USER command is:

```
USER,VHN35,VERN.
```

†At some sites, you don't need a password. You can also have different passwords for batch and interactive jobs. Refer to Volume 3, System Commands for more information.

The third card in the deck is the CHARGE command.† This command allows site personnel to keep track of the resources you use. If your site permits, you can reenter the CHARGE command later in the job to charge subsequent resources used to a different account. The CHARGE command's format is:

```
CHARGE,chargenumber,projectnumber.
```

Parameter	Description
chargenumber	A charge number site personnel assign to you.
projectnumber	A project number site personnel assign to you.

The CHARGE command parameters are order-dependent.

Example:

If you are assigned the charge number 5239 and the project number PJ325, then your CHARGE command is:

```
CHARGE,5239,PJ325.
```

If you want to use the default charge and project numbers that site personnel assigned to your user name when they authorized you to use the system, then simply omit the CHARGE command.

If you make an error on the Job, USER, or CHARGE command, your job is not allowed to enter the system.

†At some sites, you don't need the CHARGE command.

Interactive Jobs

In an interactive job, you enter the system by logging in. In the following discussion, we show the abbreviated form of the login; a complete explanation of login is in appendix C.

After dialing in and identifying your terminal, you enter the login sequence. This sequence begins with the terminal displaying:

```
WELCOME TO THE NOS SOFTWARE SYSTEM.
COPYRIGHT CONTROL DATA 1978, 198x.
yy/mm/dd.  hh.mm.ss.  terminalname
CDC NETWORK OPERATING SYSTEM.          NOS 2
FAMILY:
```

The first two lines inform you of the CDC software system and copyright. The third line is the date, time, and terminal name. The fourth line identifies the host system being used and the version of NOS. On the line beginning with FAMILY, you enter your family name, user name, password, and application separated by commas.

Example:

If your family name is NYSUB, your user name is VHN35, and your password is VERN, you enter:

```
NYSUB,VHN35,VERN,IAF
```

If your family is the default family and IAF is the default application, you can omit the family name and IAF. In this case, you enter:

```
,VHN35,VERN
```

The system responds with:

```
JSN:  jsn,NAMIAF†
READY.
```

jsn is the job sequence name the system associates with your interactive job. You use this name when referring to your job.

†The system response may differ if the CHARGE command is required for interactive jobs.

COMPLETING A TASK

For many tasks, the commands you use to complete the task and the order in which you enter them are the same for batch and interactive jobs.

Most commands have a common format. They begin with a name that describes the function of the command. For example, the REWIND command rewinds files. Usually following the name are parameters that specify file names and other options used in the operation. You must put separators between parameters and a terminator at the end to complete the command. Two of the most common separators are:

, and (

In this guide, we use the comma as the separator.

The terminator can be:

. or)

In this guide, we use the period as the terminator.

For most commands, the system ignores spaces. For example, the system treats:

REWIND, TOP .

the same as:

REWIND,TOP.

It treats:

ROUTE,OLD,DC = LP,E C=A6.

the same as:

ROUTE,OLD,DC=LP,EC=A6.

Parameters are of two types, order-dependent and order-independent. If you must have the parameters for a command in a special order, they are order-dependent; and we say so in the description of the command. When parameters are in the format:

keyword=value

the order in which you specify them usually does not matter (they are order-independent).

Many parameters have system-defined defaults; that is, if you omit the parameter, the system supplies a value. To use the default for order-dependent parameters, you omit the parameter but specify the separators. When you use the comma as the separator, this means you may have two successive commas within the command.

For example, if you are copying from the default file INPUT using the COPYBR command, you can specify:

COPYBR,,SNOW.

To use the default for order-independent parameters, you omit both the parameter and the separator.

Batch Jobs

In a batch job, all commands that complete a task or tasks must follow the Job, USER, and CHARGE commands. In the Job and USER commands, the spacing must be exactly as shown in the formats. If you add a space, the system aborts your job.

When the system is executing commands, it sometimes requires additional input. For example, if you use the program command:

```
FTN5.
```

the system assumes the FORTRAN program is located in your job file. You must put any input needed during the execution after all of the commands. You separate commands from input by a multipunched 7/8/9 card. A 7/8/9 card has rows 7, 8, and 9 punched in column 1. This card represents an end-of-record, which we describe later.

If more than one command uses input, you must separate groups of input by a 7/8/9 card. If you do have more than one input, make sure you put them in the order that the system will need them. For example, a job that requires a program as the first input and data as the second input must have the structure shown in figure 2-2.

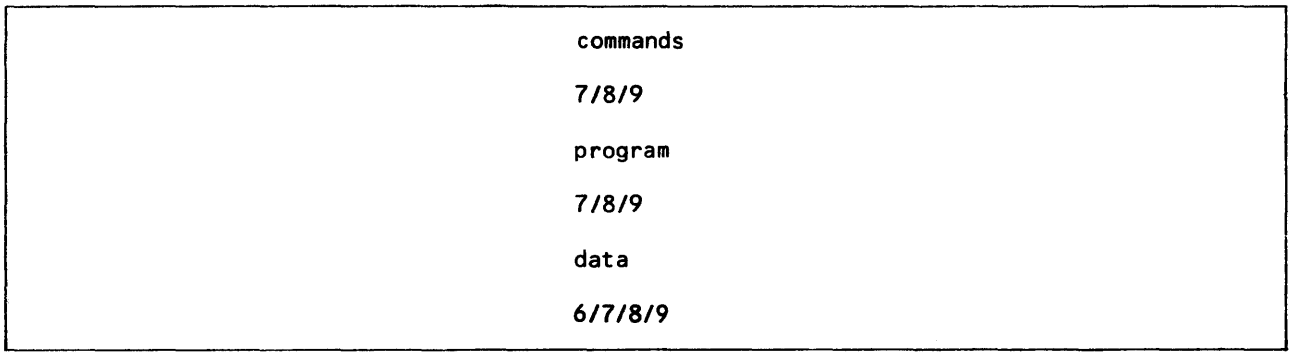


Figure 2-2. Structuring Input

Interactive Jobs

In an interactive job, you usually enter one command and then wait for the system to respond. When entering commands interactively, you can omit the terminator. The system assumes that a command is complete when you enter a carriage return. This is not true of commands in procedures and submit files (we describe these files later). Although you can create these files interactively, the files are processed as though they are part of a batch job.

When the system wants input from the terminal, it displays a prompt. You then enter the appropriate response. The system processes your response and returns with a prompt either for more input or for another command. For some operations, the system does not accept input that you enter from the terminal. For example, with the FTN5 command (a call to the FORTRAN 5 compiler), the FORTRAN program must be in a file. You must name the file in the I=program parameter of that command.

If there is any output from individual operations, the system returns it to the terminal after you enter the command. Consequently, you know immediately whether or not the command produced the results you wanted.

LEAVING THE SYSTEM

Leaving the system is accomplished differently for batch jobs entered as a card deck and interactive jobs.

Batch Jobs

In a batch job submitted on a card deck, you signify that your job is finished by putting a multipunched 6/7/8/9 card at the end of the deck (rows 6, 7, 8, and 9 punched in column 1). This card signifies the end of information and terminates the job deck.

The system prints all output from the job on a central site line printer, unless you specify otherwise. The output includes a history of the commands processed during the job and messages relating to those commands. If errors occur in processing, a message to that effect appears in this listing. The listing is called the user's dayfile.

Interactive Jobs

In an interactive job, you finish a job when you log out by entering a logout command, such as BYE, in response to a system prompt. The system then displays:

```
UN=username  LOG OFF      hh.mm.ss.
JSN=jsn       SRU-S           s.sss.
CHARACTERS=    1.234KCHS
IAF     CONNECT TIME hh.mm.ss.
LOGGED OUT.

HOST DISCONNECTED    CONTROL CHARACTER=ct
ENTER INPUT TO CONNECT TO HOST
```

and disconnects the terminal. username is your user name. jsn is the job sequence name the system assigned to this session. The first hh.mm.ss. is the time of logout. s.sss is a measure of the system resources you used while connected to IAF. The second hh.mm.ss. tells how long you were connected to IAF. ct is the key or keys on the terminal that can be used to change terminal definitions.

SUBMIT FILES

You can create a special file, called a submit file, that contains all the commands and data necessary for a job. You can enter the submit file for processing through a batch job, but it is most commonly done interactively.

The submit file contains images of cards in a batch job (including terminators). The system treats the submit file as it would treat a batch job. The file that the system executes has the same structure as a batch job; however, the file you create can contain SUBMIT directives that help you structure the file interactively. When you enter the file for processing, the system changes the file according to the directives before executing it.

You submit the file for processing with the SUBMIT command. Its format is:

```
SUBMIT,lfn,q.
```

Parameter	Description
lfn	The name of the local file to be submitted for processing as a batch job.
q	Tells the system what to do with the job output. q can be one of:

q	Description
BC	Output is printed at the central site. BC is the default when the SUBMIT command is executed from a batch job.
NO	Output is not printed. NO is the default when you enter the SUBMIT command interactively.
TO	Output is held in the system wait queue either until you access it or until it is purged. The amount of time it remains in the system before it is automatically purged is determined by site personnel. To access the output, you can enter: `QGET,JSN=jsn,DC=WT.` jsn is the job sequence name associated with the submit file and becomes the name of the file containing the output. Refer to Volume 3, System Commands, for more information.
RB=username	Output is printed at a remote batch terminal logged in under user name username. If you specify RB only, the user name in effect for the submitting job is assumed.

When the system processes the SUBMIT command, it returns the message:

```
hh.mm.ss. SUBMIT COMPLETE.  JSN IS jsn
```

hh.mm.ss is the time the command was processed; jsn is the job name assigned to the submit file. When you see this message, you know the file has been submitted for execution.

Although you can use the SUBMIT command in both batch and interactive jobs, for the following discussion we assume you are using an interactive terminal.

STRUCTURING A SUBMIT FILE

When you create a submit file, you can use directives to help you structure it so that it looks like a batch job when it is executed. When you submit the file for processing, the system reformats it according to your directives and then queues it to await execution.

The following directives allow you to establish batch job divisions within the file and to either retain or delete line numbers. Additional directives are described in Volume 3, System Commands.

Directive	Description
/JOB	Indicates that the system is to reformat the submit file. Reformatting directives are not processed unless /JOB is the first line in the submit file. Even if you do not use any other directives, you must use /JOB if the submit file has line numbers.
/EOF	Inserts an end-of-file (6/7/9 card in a batch job) at this point in the submit file. Use this directive to create files within a file (a multifile file; multifile files are discussed in section 3).
/EOR	Inserts an end-of-record (7/8/9 card in a batch job) at this point in the submit file.
/SEQ	Indicates that line numbers are to be removed from all lines following this line. SEQ is in effect unless you disable it by using NOSEQ.
/NOSEQ	Indicates that line numbers are not to be removed from all lines following this line. Use this directive with a BASIC program, in which line numbers are required, or with numbered data, in which each line begins with a number that should not be interpreted as a line number.
/USER	Inserts a USER command specifying the user name, password, and family name in effect for the calling job. The /USER directive allows you to avoid inserting this information in the submit file.
/CHARGE	Inserts a CHARGE command that specifies the charge and project numbers currently in effect for the calling job.

Figure 2-3 shows the reformatting of a submit file. The left side of the figure shows the file as the user typed it at the terminal; the right side shows the reformatted file that is executed.

Original Submit File	Reformatted Submit File
00010 /JOB	
00020 JONES.	JONES.
00030 /USER	USER,username,password,familyname.
00040 BASIC.	BASIC.
00050 /EOR	end-of-record
00060 /NOSEQ	
00070 REM PROGRAM TEST	00080 REM PROGRAM TEST
.	.
.	.
.	.
(BASIC source program)	(BASIC source program)
.	.
.	.
.	.
00290 END	00290 END
00300 /SEQ	
00310 /EOR	end-of-record
00320 " FIRST LINE"	" FIRST LINE"
.	.
.	.
.	.
(input for BASIC program)	(input for BASIC program)
.	.
.	.
.	.
00590 " END"	" END"

Figure 2-3. System Reformatting of a Submit File

CREATING A SUBMIT FILE

The most common ways of creating a submit file interactively are:

- Using the AUTO command.
- Using the TEXT command.
- Using a text editor, such as the Full Screen Editor (FSE).

The TEXT command is described in Volume 1, Introduction to Interactive Usage; we describe FSE later in this guide.

The AUTO command automatically generates line numbers. If you enter:

```
AUTO.
```

the system starts generating line numbers at 00100 with an increment value of 10. To cancel the AUTO command, enter the termination sequence right after the generated line number. For most ASCII terminals the termination sequence is CTRL/T followed by a carriage return (while holding down the control key, press T, then release both keys and press the carriage return). Since this may vary, consult site personnel if this does not work at your terminal.

In figure 2-4, the user creates a submit file with the AUTO command.

Terminal Session	Description
/new,demo/nd	The user creates a file named DEMO which will contain the submit file.
/auto	The user enters the AUTO command to generate line numbers.
00100 /job 00110 myjob. 00120 /user 00130 copybf,,newlist. 00140 replace,newlist. 00150 /eor 00160 item one 00170 item two . . . 00320 item last 00330 termination sequence	The system generates the line numbers; the user completes the lines and enters carriage returns.
TERMINATED	The user terminates automatic line numbering by entering the termination sequence (usually CTRL/T).
/submit,demo,bc	The user submits file DEMO for processing. The output is printed at the central site.
08.42.36. SUBMIT COMPLETE.	JSN IS ARFR

Figure 2-4. Creating a Submit File

If you make a mistake on a line, you can exit AUTO mode and enter the line number and a new line to correct the mistake. NOS replaces the old line with the new line. In figure 2-5, the user corrects a mistake made while creating a submit file. When correcting mistakes while using automatic line numbering, remember that a new line number is generated each time you press the carriage return. Therefore, you should cancel automatic line numbering before attempting to correct a line other than the last line you entered. Otherwise, you may change lines you did not intend to change.

Terminal Session	Description
/ new,demo2	The user creates file DEMO2.
/ auto	The user enters the AUTO command.
00100 / job 00110 yourjob.	The system generates line numbers; the user enters lines.
00120 use .	The user makes a mistake, but not realizing it, continues entering lines.
00240 termination sequence *TERMINATED*	The user sees the mistake in line 00120 and terminates the automatic line numbering by entering the termination sequence (usually CTRL/T).
/00120 / user	The user enters the entire line to correct the mistake in line 00120.
/ auto,240	The user enters the AUTO command to begin automatic line numbering at line 00240.
00240 / noseq .	The system supplies line numbers starting at 00240, and the user continues entering lines into the file.

Figure 2-5. Correcting Mistakes in Submit Files

MONITORING A SUBMIT FILE

You can monitor the progress of your submit files with the ENQUIRE command. Its format is:

```
ENQUIRE,JSN=jsn.
```

jsn is the job sequence name assigned to the submit file you want to monitor (the system displays this name at the terminal after you enter the SUBMIT command).

In response to this command, the system displays the status of your file, that is, whether it is being executed, printed, and so forth. In figure 2-6, the user monitors a submit file with this format of the ENQUIRE command.

Terminal Session	Description
/submit,subfile,bc	The user submits a file called SUBFILE to be processed. The output is to be printed at the local site.
11.33.13 SUBMIT COMPLETE. JSN IS AMAC	The system displays the job sequence name assigned to file SUBFILE (AMAC).
/enquire,jsn=amac	The user enters the ENQUIRE command to determine the status of the submit file.
AMAC.B. .BC. .PRINT QUEUE UJN=AVAQ PRU LENGTH= 2. DC=LP. FC= . ID= 0. EC= .	The system responds that job AMAC has been executed and is waiting to be printed.
/enquire,jsn=amac	After a moment, the user enters the ENQUIRE command again.
AMAC NOT FOUND.	The system responds that job AMAC is either being printed or has finished printing.

Figure 2-6. Determining the Status of a Submit File

To determine the status of more than one of your submit files, you can enter:

ENQUIRE,JSN.

In figure 2-7, the user monitors two submit files.

Terminal Session	Description
/ submit,file1,bc	The user submits file FILE1, whose output is to be printed at the central site.
15.32.12 SUBMIT COMPLETE. JSN IS BCCB	The system displays the job sequence name assigned to FILE1 (BCCB).
/ submit,file2,no	The user submits file FILE2, which is to be executed only; no output is to be printed.
15.34.06 SUBMIT COMPLETE. JSN IS BCDD	The system displays the job sequence name assigned to FILE2 (BCDD).
/ enquire,jsn	The user enters the ENQUIRE command to determine the status of both files.
JSN SC CS DS LID STATUS JSN SC CS DS LID STATUS BCAA.T.ON.BC. .EXECUTING BCCB.B. .BC. .PRINT QUEUE	The system responds that the user's interactive job (BCAA) is executing and that BCCB is waiting to be printed. Job BCDD is not shown because it has executed.

Figure 2-7. Determining the Status of More Than One Submit File

The system returns the job sequence name and the status of all jobs associated with your user name. Information is also returned giving the job service class (SC), the connection status (CS), the default destination (DS), and the logical identifier (LID) of the mainframe. If you want more information on these fields, refer to Volume 3, System Commands.

USING SYSTEM RESOURCES

When site personnel authorize you to use the system, they decide which system resources you can use, and set limits on the quantity you can use of any particular resource. For example, you may be allowed to use magnetic tape units but no more than two at one time; you may be allowed to run a job but to use no more than an allotted number of system resource units (SRUs).

To find out what your allocations and limits are, you can enter the LIMITS command as follows:

```
LIMITS.
```

The system displays a list of some of your allocations and limits. Included in this list are:

- The maximum number of:
  - Magnetic tape units you can use at one time.
  - Removable disk packs you can use.
  - Central memory words you can use (maximum field length).
  - Seconds that a command can take to be processed (CPU seconds allowed).
  - Permanent files you can have.
- The default charge number and project number for your user name (if your site uses them).
- The subsystem you are automatically assigned to when you log in.
- What you can do within a job (for example, whether you can change your passwords or create permanent files).

Figure 2-8 shows an example of the LIMITS command output.

```
MAXIMUM NUMBER OF -
 MAGNETIC TAPE UNITS THAT MAY BE ASSIGNED.....          3
 REMOVABLE AUXILIARY DEVICES THAT MAY BE ASSIGNED       2
 CPU SECONDS ALLOWED FOR EACH JOB STEP..... UNLIMITED
 CENTRAL MEMORY WORDS ALLOWED/100B.....          2037B
 EXTENDED MEMORY WORDS ALLOWED/1000B.....          0B
 JOBS THAT CAN BE DETACHED.....          3
 DEFERRED BATCH FILES.....          8
 PERMANENT FILES ALLOWED IN YOUR CATALOG..... UNLIMITED
 PRUS AVAILABLE FOR ANY ONE INDIRECT FILE.....        192
 PRUS AVAILABLE FOR ANY ONE DIRECT FILE.....       1024
 PRUS AVAILABLE FOR ALL INDIRECT FILES.....       4096
 PRUS AVAILABLE PER JOB.....      66048
 MESSAGES THAT MAY BE ISSUED TO THE DAYFILES.....       1008
 BATCH COMMANDS THAT MAY BE PROCESSED..... UNLIMITED
 CARDS THAT MAY BE PUNCHED PER FILE..... UNLIMITED
 LINES THAT MAY BE PLOTTED PER FILE.....        512
 LINES THAT MAY BE PRINTED PER FILE..... UNLIMITED
 SRUS ALLOWED PER JOB..... UNLIMITED

OTHER CHARACTERISTICS -
 USER INDEX.....          3360B
 DEFAULT CHARGE NUMBER ASSIGNED.....          2222
 DEFAULT PROJECT NUMBER ASSIGNED.....       666N666
 SYSTEM PROLOGUE.....          2B
 USER PROLOGUE.....        PROLOG
 INTERACTIVE SUBSYSTEM.....         BATCH
 INTERACTIVE CHARACTER SET MODE.....        NORMAL

THE FOLLOWING ARE VALID USER PERMISSIONS -
 CHANGE YOUR PASSWORD
 CREATE DIRECT ACCESS FILES
 CREATE INDIRECT ACCESS FILES
 ACCESS/CREATE LIBRARY FILES
 ASSIGN MAGNETIC TAPES
 ACCESS OR CREATE FILES ON AUXILIARY DEVICES
 TRANSFER PERMANENT FILES BETWEEN HOSTS
 TRANSFER QUEUED FILES BETWEEN HOSTS
 SPECIFY LID-S ON JOB AND *ROUTE* COMMANDS
 CHARGE NOT RESTRICTED TO DEFAULT
 DEFINE NON-RANDOMIZED LOGIN PASSWORD

THE FOLLOWING ARE VALID NETWORK APPLICATIONS -
 INTERACTIVE FACILITY (IAF)
 REMOTE BATCH FACILITY (RBF)
 TRANSACTION FACILITY (TAF)
 TERMINAL VERIFICATION FACILITY (TVF)

THE FOLLOWING ARE VALID SHELL PERMISSIONS -
 PROCESS COMMANDS OUTSIDE CCL PROCEDURE
 SYSTEM LIBRARY LOAD OF SHELL PROGRAM

THE FOLLOWING ARE VALID SECURITY PERMISSIONS -
 WRITE UNLABELED MAGNETIC TAPES

THE FOLLOWING ARE VALID SECURITY ACCESS LEVELS -
 ACCESS LEVEL LVL0

THE FOLLOWING ARE ALLOWED SERVICE CLASSES -
 BATCH (BC)
 REMOTE BATCH (RB)
 INTERACTIVE (TS)
 DETACHED INTERACTIVE (DI)

THE USER DEFAULT SERVICE CLASS
 FOR BATCH ORIGIN IS..... BC
 FOR REMOTE BATCH ORIGIN IS..... RB
 FOR INTERACTIVE ORIGIN IS..... TS
INQUIRY COMPLETE.
```

Figure 2-8. Resource Allocations and Limits

The fact that you are allocated certain resources does not mean you automatically get the maximum allowed for that resource. This is usually true of the amount of time allowed in the CPU and the number of SRUs used in processing each command. For the average job, the resources allocated are usually adequate for the job. However, if you reach a time limit or SRU limit, you can increase these resources either by entering the appropriate response in an interactive job or by using the SETTL or SETJSL command in a batch job. If increasing these resources does not allow your job to finish, check the job for errors. If there are no errors, change the job so that it uses less resources. Time limit and SRU limit are discussed in detail in Volume 3, System Commands (refer to the SETTL and SETJSL commands). In figure 2-9 the user corrects a time limit problem.

Terminal Session	Description
/get,time /ftn5,i=time,l=out,go *TIME LIMIT* ENTER T TO CONTINUE OR CR KEY TO STOP:	The user gets a FORTRAN program and executes it.
t	The user increases the amount of time she or he can use.
COMPILATION COMPLETE.	The program completes.

Figure 2-9. Time Limit

If you are interested in finding out what your actual resources are for a particular job, enter the following ENQUIRE command.

```
ENQUIRE,OP=A.
```

This command gives you added information about your job as well as listing the resources available to you at that point in the job.

FILES 3

A file is a collection of information that you and the system refer to by a single name. Whenever you enter a program or text into the system, either from a batch job or interactively, you put the information in a file. A file begins with a beginning-of-information (BOI), supplied by the system, and ends with an end-of-information (EOI), supplied by either you or the system. A file can be empty or it can be as large as your resource allowances and limits permit (resources are explained under Using System Resources in section 2).

You can divide a file into smaller parts. This allows you easy access to part of a file and also marks boundaries between different areas within a file. Each part is called a system logical record.

You can also put several files together into a multifile file. This allows you to reference more than one file by a single name. For example, student rosters from a number of classes can be put together into one departmental roster (figure 3-1).

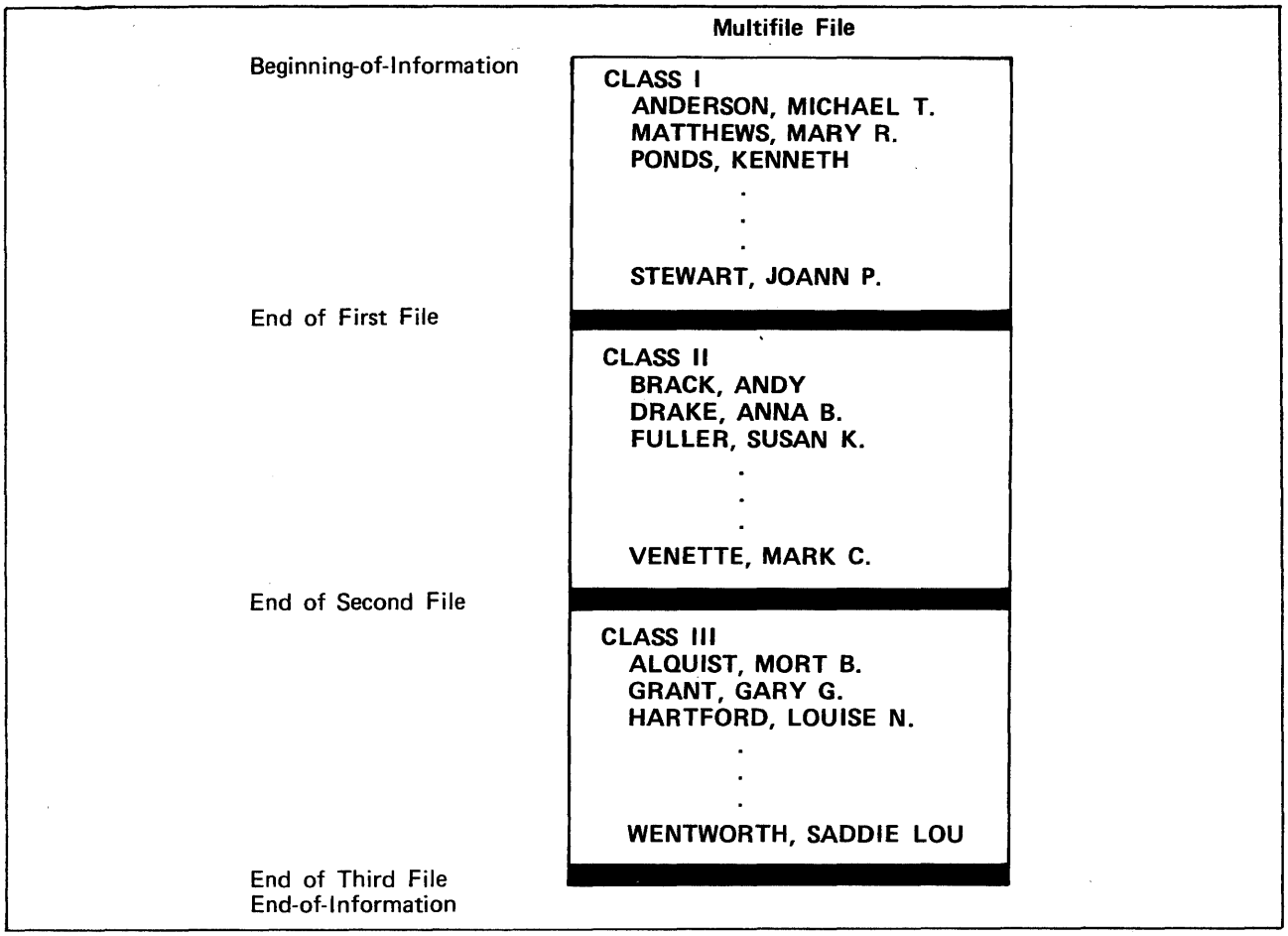


Figure 3-1. Example of a Multifile File

FILE NAMES

Each file must have a unique name. This name can be any combination, from one through seven characters, of the following alphanumeric characters:

A through Z and 0 through 9

When you create a file, you give it a name. Although NOS accepts a name starting with a digit, some products (for example, COBOL) do not allow a file name to begin this way. To be safe, use a letter as the first character in file names.

In the case of a multifile file, only the multifile file has a name. The files within the multifile file cannot be referenced separately and therefore do not need names.

Several file names are either reserved for system use or have special significance to the system. You should avoid using these names when naming your own files. The NOS file names with special significance include:

INPUT	P8	SCR3
OUTPUT	SCR	SCR4
PUNCH	SCR1	ZZxxxxx
PUNCHB	SCR2	

ZZxxxxx represents any file whose name begins with ZZ. Products used with NOS also reserve file names for internal use. Check the reference manual for each product you use to obtain additional reserved file names.

If you use one of the special names, either you get the message:

RESERVED FILE NAME.

or your job may not work as you had intended.

LOGICAL FILE STRUCTURE

Logical file structure is the order you apply to a file. For example, when you create a file that contains both a program and data for that program, you place the program first and then the data, with a separator between the two.

NOS uses a file structure incorporating records and files within files. To create this file structure you can use FSE, NOS commands, or NOS delimiters (explained later). At times you may want to use more advanced file structures, which involve using the CDC CYBER Record Manager (CRM). For more information on advanced file structures, refer to the reference manual for the language you are using or to the CYBER Record Manager reference manuals.

NOTE

CRM recognizes internal file structures other than those used with NOS and, therefore, CRM file structure is not totally compatible with NOS commands. Because of the differences, we do not recommend manipulation of CRM files using NOS commands.

Example:

> The complete file contains the names of employees in the manufacturing division of a company. It is a multifile file, which is composed of smaller files, each file containing the names of employees on a particular shift. These smaller files are divided into units called system logical records (explained later). Each system logical record has the names of employees that work on a particular assembly line.

Although each file has a specific structure, you need not be concerned with this structure for much of your work on the system. If you always use the same set of commands when working with a file, its structure stays constant.

In describing the commands in sections 4, 5, and 6, we show only the file structure that is relevant to the discussion of the command. If you are working with files that have a complex file structure and are interested in changing parts of that structure, you should read Volume 3, System Commands.

RECORDS

A file consists of zero or more system logical records. A record can be either fixed or variable in length. The contents of a record do not depend on where it is stored; that is, a record on punched cards contains the same information that it would contain if it were on mass storage. In this guide, the word record refers to a system logical record.

A record begins at one of three places: at BOI, after end-of-record (EOR), or after end-of-file (EOF). A record ends with EOR. Later in this section we explain BOI, EOR, and EOF more completely.

A record's length is measured in physical record units (PRUs). A PRU is the smallest recording unit provided by a device. The size of a PRU varies among devices (refer to Physical File Structure in Volume 3, System Commands, if you want more information). The number of PRUs you are allowed to use is given in your resource allocations and limits (discussed in section 2). The number of PRUs you are using is shown when you list the files you have on the system (we discuss this in section 5).

STRUCTURING A FILE

You can structure a file using system commands (such as the copy commands or the Full Screen Editor), a high-level language (such as FORTRAN), or delimiters. We discuss the copy commands in section 4. The reference manual for the language you are using shows how the language helps you structure files. If you are using RBF, consult the Remote Batch Facility Reference Manual for information on delimiters. A discussion of the use of delimiters follows.

Delimiters specify the end of a record or the end of a file. For card decks, you add delimiters to a file as multipunched cards. For interactive jobs, you can add delimiters to the file when it is created or edited.

Beginning-of-Information (BOI)

The beginning of a file is called BOI. The system establishes and keeps track of BOI for each file. There is no actual mark on the file to signify BOI. When the system positions the file at BOI, it is positioned in front of anything you have put in the file.

End-of-Record (EOR)

You insert EOR in a file to specify that all the information between either BOI or the previous EOR or EOF and the inserted EOR is one record. For a local batch job, EOR is a card with rows 7, 8, and 9 punched in column 1 (refer to figure 3-2). For a submit file (explained in section 2), EOR is the directive /EOR. You can also use FSE to insert EORs, enclosed in parentheses, wherever needed in the file.

End-of-File (EOF)

You insert EOF in a sequence of information to specify that all information between either BOI or the previous EOF and the inserted EOF is one file. For a local batch job, EOF is a card with rows 6, 7, and 9 punched in column 1 (refer to figure 3-2). For a submit file, EOF is the directive /EOF. You can also use FSE to insert EOFs, enclosed in parentheses, wherever needed in the file.

End-of-Information (EOI)

Either you or the system add EOI to a sequence of information to specify that all the information between BOI and EOI is to be referenced by the name assigned to the file or multi-file file. For a local batch job and some remote batch jobs, an EOI card is a card with rows 6, 7, 8, and 9 punched in column 1 (refer to figure 3-2). It must be the last card in the deck. For a submit file, the system adds EOI to the end of the information.

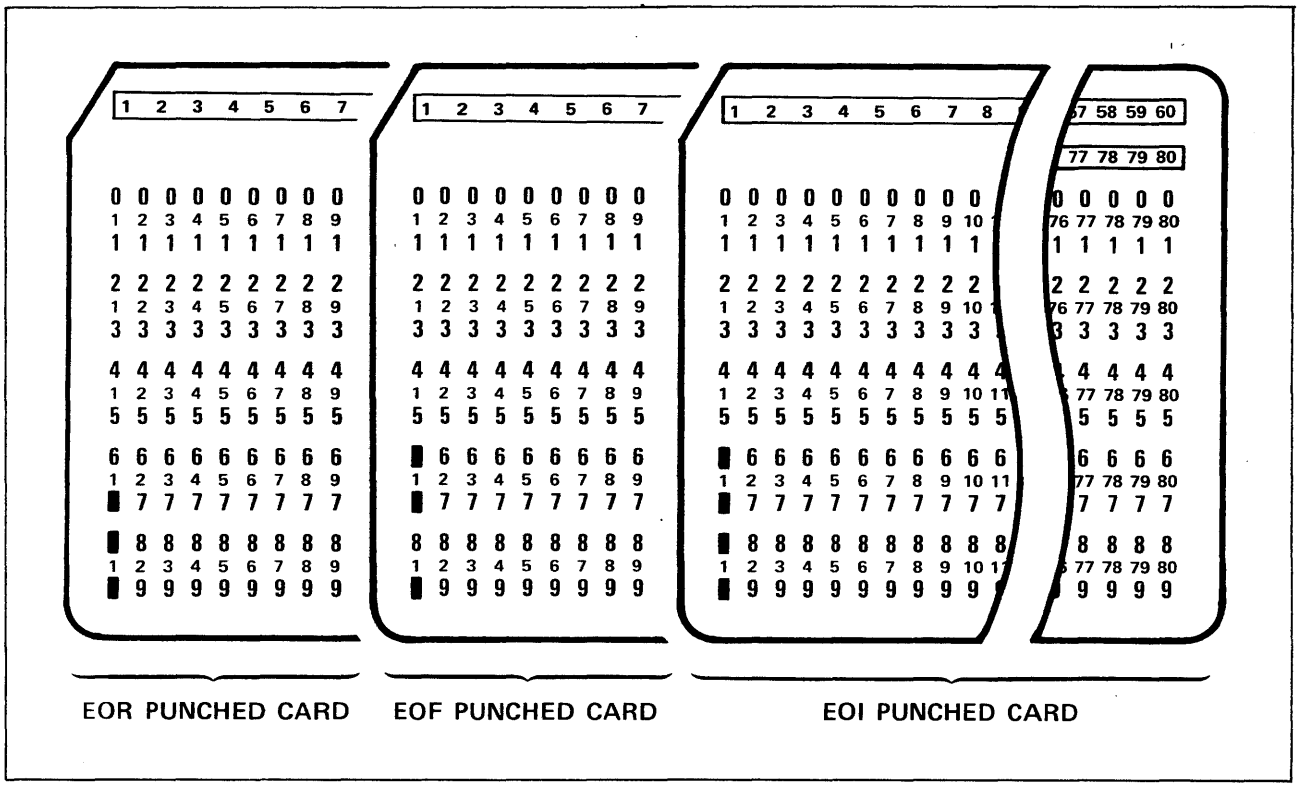


Figure 3-2. EOR, EOF, and EOI Representations on Punched Cards

FILES INPUT AND OUTPUT

The system creates files named INPUT and OUTPUT each time you use the system interactively or enter a batch job. The following discussion defines these files and describes how the system uses them.

FILE INPUT

File INPUT contains information used during your job. The system treats file INPUT differently for interactive and batch jobs.

Interactive File INPUT

For an interactive job, your terminal keyboard is considered file INPUT. Because the system usually processes each command right after you enter it, commands do not remain in file INPUT very long.

For many commands, when either you do not specify the file to be used or you specify file INPUT, the system asks you to enter the data from the terminal. Depending on the command, the system may accept one line of data (as in the COPYBR command) or many lines of data (as in the MODIFY and LIBEDIT commands). A line of data ends when you enter the line terminator (usually called the carriage return). In the following terminal session, the COPYBR commands are used for copying one line of data into files BIRD and DOG.

Terminal Session	Description
/new,bird /copybr,,bird	The user enters a COPYBR command that leaves out the name of the file the system is copying; therefore, file INPUT is copied.
? record 1 of file bird COPY COMPLETE.	The system asks for input and the user enters a line of data followed by a carriage return.
/copybr,,dog	The user again enters a COPYBR command.
? record 1 of file dog COPY COMPLETE.	The system asks for input and the user enters a line of data followed by a carriage return.

Batch File INPUT

For a batch job, file INPUT is the entire job you submit for processing. The first record of file INPUT contains the commands and is used by the system. The remaining records and files contain in sequential order any data that is to be used with the commands. To reference this data in commands, you either enter INPUT as the name of the file to be used or, if the default file for the command is INPUT, omit the file name. The example in figure 3-3 shows a batch job that has three records: the command record and two records containing data.

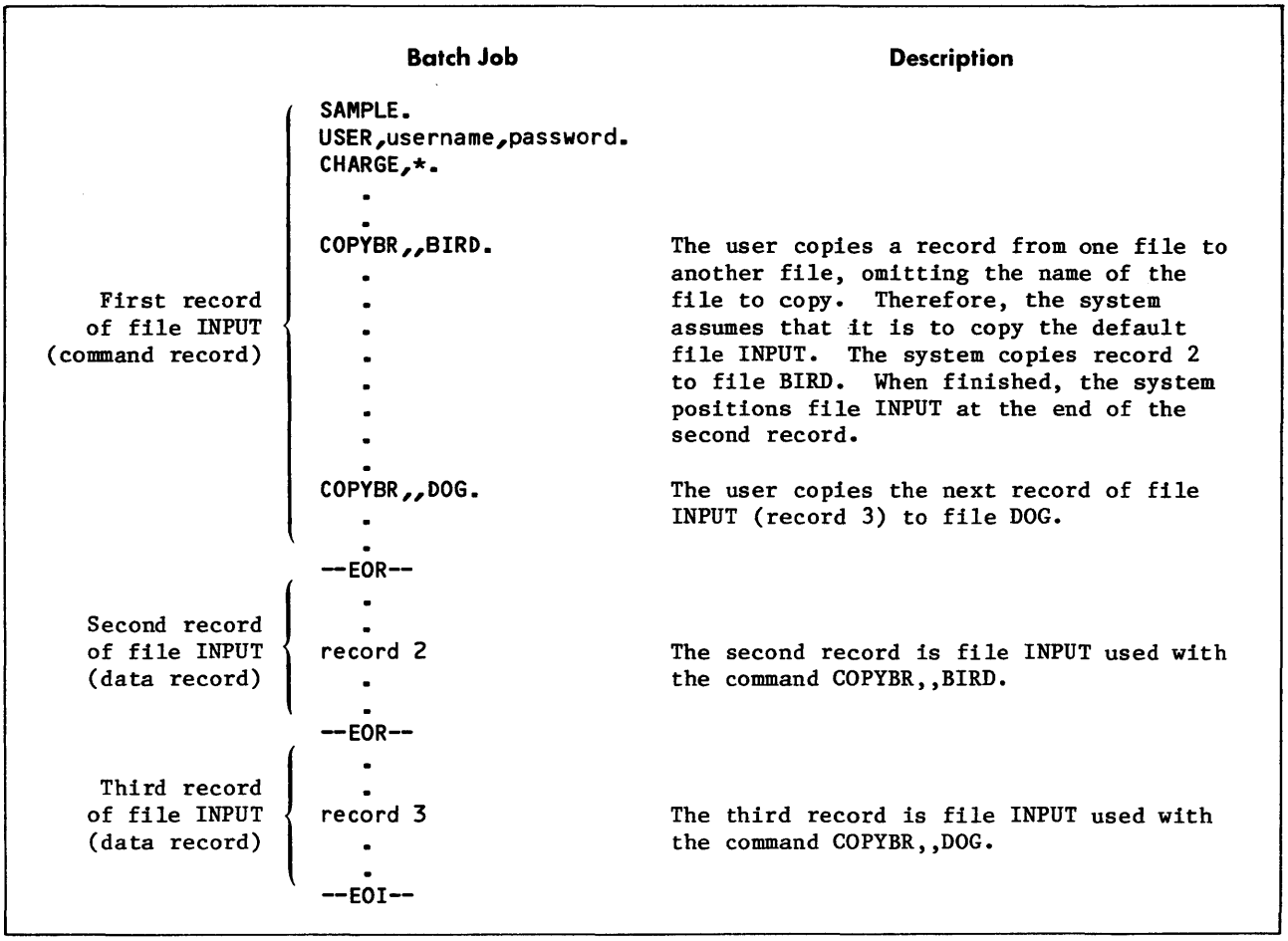

Batch Job
Description
SAMPLE.
USER,username,password.
CHARGE,*.
COPYBR,,BIRD.
The user copies a record from one file to another file, omitting the name of the file to copy. Therefore, the system assumes that it is to copy the default file INPUT. The system copies record 2 to file BIRD. When finished, the system positions file INPUT at the end of the second record.
COPYBR,,DOG.
The user copies the next record of file INPUT (record 3) to file DOG.
First record of file INPUT (command record)
--EOR--
record 2
The second record is file INPUT used with the command COPYBR,,BIRD.
Second record of file INPUT (data record)
--EOR--
record 3
The third record is file INPUT used with the command COPYBR,,DOG.
Third record of file INPUT (data record)
--EOI--

Figure 3-3. File INPUT for Batch Jobs

FILE OUTPUT

File OUTPUT contains all data resulting from program execution and, for most commands, the data resulting from command processing when you do not specify a file name to contain the data. What happens to the file depends on the type of job to which it is associated.

- In batch jobs, the system prints file OUTPUT on a central site line printer unless you specify otherwise.
- In interactive jobs, the system displays file OUTPUT at the terminal right after it processes a command.

The information displayed at the terminal is not always the same as what is printed on the line printer. Refer to individual command descriptions to determine what is printed for a particular job. In some cases file OUTPUT is not formatted correctly to be printed at the terminal. The utilities LIST80 and LO72 can reformat files for terminal output. If your files need to be reformatted, refer to Volume 3, System Commands for more detailed information.

Files you name PUNCH, PUNCHB, or P8 are automatically queued by the system to be punched when you terminate your job.

ROUTING YOUR FILES

To have files printed at the central site you can use the ROUTE command. You can also use the ROUTE command to submit a batch job for processing. The job you route must be structured like a batch job. ROUTE does not have directives for structuring as SUBMIT does.

The ROUTE command's format is:

```
ROUTE,lfn,p1,p2,...,pn.
```

Parameter	Description
lfn	Name of the file to route; it cannot be a primary file or a direct access file (discussed in section 5).
pi	Optional parameters. The following parameters are frequently used.

pi	Description
DC=xx	Disposition code. This parameter determines where the file goes in the system.

xx	Description
LP	Print on any printer.
IN	Queue for input.
TO	Queue for input, and place output in the wait queue.
WT	Queue to wait.

Parameter	Description

p_i	Description
EC=xx	External characteristics for files to be printed. If you are not sure what characteristics your file has, check appendix A. It describes the character sets you can use.

xx	Description
A6	ASCII graphic 63/64-character set used for printing files. The file must be a 6-bit display code file. This parameter causes your file to be printed in uppercase.
A9	ASCII graphic 95-character set used for printing files. The file must be a 7-bit ASCII code file. This parameter causes your file to be printed in uppercase and lowercase characters.
B6	CDC graphic 63/64-character set used for printing files. The file must be a 6-bit display code file.

Files that are to be printed at the central site need to be formatted to use the carriage control characters for the printer. These characters are located in column 1 of the file and are not printed. If you have a file that is not formatted correctly for printing at the central site, you can use the COPYSBF command to reformat it (refer to section 4).

To print at the central site a file that was created interactively using uppercase and lowercase characters, you must convert the file to 7-bit ASCII code before using the ROUTE command. The FCOPY command performs the conversion. Its format is:

FCOPY,P=lfn_1,N=lfn_2,R.

lfn_1 is the file to be converted and lfn_2 is the file that contains the converted data. R specifies that files lfn_1 and lfn_2 are rewound (set to beginning-of-information) before and after copying. If you omit R, neither file is rewound. Figure 3-4 shows an example of having the central site print a file in uppercase and lowercase characters.

```
/get,box
/fcopy,p=box,n=present,r
 FCOPY COMPLETE.
/route,present,dc=lp,ec=a9
  ROUTE COMPLETE.  JSN IS ACPA.
```

Figure 3-4. Printing an Uppercase and Lowercase File

TEMPORARY AND PERMANENT FILES

Every one of your files is either a temporary file or a permanent file.

- Temporary files are files the are assigned to your job and are not saved on permanent storage. Temporary files are no longer accessible to you or to another user after the interactive or batch job that creates them is finished.

- Permanent files are files that are saved on disk. Permanent files are saved from one job to the next.

A local file is any file that is assigned to your job. Most local files that you create during interactive and batch jobs are temporary files. We discuss local files in section 4.

NOS supports two types of permanent files, indirect access and direct access. When you access an indirect access file, the system creates a temporary copy of the file for use during your job. The changes you make are made to the copy only; the permanent file is unchanged. No temporary copy is made for direct access files that you access; you use the permanent file itself. Those users who have authorization from site personnel can create new permanent files and make temporary files permanent. We discuss the different types of permanent files in section 5.

LOCAL FILES 4

All files associated with your job, whether the job is interactive or batch, are local files. They can be files you created during the job and permanent files you retrieved. During a job you can create, position, rename, and release local files. In this section we explain some common ways of performing these things; we do not include all the possible ways nor all the parameters for each command. If you want additional information on a command, refer to Volume 3, System Commands.

CREATING LOCAL FILES

You create a local file when you:

- Create either a new temporary file (the NEW command) or a new direct access permanent file (the DEFINE command is described in section 5).
- Make a temporary copy of an indirect access permanent file (the OLD or GET command).
- Attach a previously created direct access permanent file to your job (the ATTACH command).
- Assign a tape file to your job (the LABEL command).
- Run a program that creates one or more files (refer to the reference manual for the language you are using).
- Use the name of a file for the first time in one of the copying commands.
- Create a file using an editor, such as FSE (refer to section 9).

You can designate one of your local files as the primary file. The primary file is automatically rewound before each operation and is the default file when you do not specify a file name for operations (such as saving and replacing files). The system allows you to have only one primary file at a time. The most common way to designate a primary file is to use the NEW or OLD command.

While reading the following descriptions of the NEW, OLD, GET, and ATTACH commands, refer to figure 4-1, which shows the use of the individual commands and their interactions.

NEW COMMAND

The NEW command creates an empty local file and designates it the primary file. The information placed in this file is not retained when your job is finished unless you save it during the job. The NEW command's formats are:

NEW,lfn.

This command creates a primary file named lfn and releases all your other local files that do not have a special no-auto-drop status (explained under Releasing Local Files).

NEW,lfn/ND.

This command creates a primary file named lfn but does not release your other local files. If you had a primary file before entering the NEW command, that file becomes a nonprimary local file.

OLD COMMAND

The OLD command makes a local copy of an indirect access permanent file and designates it the primary file. Any changes you make are made to the copy only; the permanent file is unchanged. The OLD command's formats are:

OLD,lfn=pfn.

This command makes a copy named lfn of a permanent file named pfn. If you want lfn and pfn to be the same, you can specify OLD,pfn. The system releases all your other local files that do not have a special no-auto-drop status.

OLD,lfn=pfn/ND.

This command makes a copy named lfn of a permanent file named pfn, but does not release your other local files. If you had a primary file before entering the OLD command, that file becomes a nonprimary file.

GET COMMAND

The GET command makes a local copy of an indirect access permanent file for use as a nonprimary file. Any changes you make are made on the copy only; the permanent file is unchanged. The GET command's format is:

GET,lfn=pfn.

This command makes a copy named lfn of the permanent file named pfn. If you want lfn and pfn to be the same, you can specify GET,pfn. The system does not release any of your local files unless its file name is the same as that specified by lfn.

ATTACH COMMAND

The ATTACH command associates a direct access permanent file with your job for use as a nonprimary file. You access the permanent file itself, rather than a copy of the permanent file. Therefore, any changes you make are made on the permanent file. The ATTACH command's format is:

ATTACH,lfn=pfn

This command associates file name lfn with the permanent file named pfn while it is attached to your job. If you want lfn and pfn to be the same, you can specify ATTACH,pfn. The system does not release any of your other local files unless another file name is the same as that specified by lfn.

Sequence of User Entries	Local Files		Comments
	Primary	Nonprimary	
NEW,BABY.	BABY		Creates new file BABY and designates it the primary file.
GET,SISTER.	BABY	SISTER	Makes a copy of file SISTER; does not release file BABY.
ATTACH,TOYS=GAME.	BABY	SISTER TOYS	Attaches direct access file GAME and names it TOYS; does not release other local files.
OLD,BROTHER.	BROTHER		Makes a copy of file BROTHER; releases all other local files.
ATTACH,FATHER.	BROTHER	FATHER	Attaches direct access permanent file FATHER; does not release file BROTHER.
GET,MOTHER.	BROTHER	FATHER MOTHER	Makes a copy of indirect access permanent file MOTHER; does not release other local files.
NEW,STORY/ND.	STORY	FATHER MOTHER BROTHER	Creates new file STORY and designates it the primary file; does not release other local files. File BROTHER becomes a nonprimary file.

Figure 4-1. The NEW, OLD, GET, and ATTACH Commands

COPYING COMMANDS

The copying commands make copies of local files and records. Descriptions of the parameters used in copying commands follow.

Parameter	Description
lfn_1	The name of the local file from which the system copies. The default is file INPUT.
lfn_2	The name of the local file to which the system copies. The default is file OUTPUT. If the file does not exist, the system creates a new local file with the name lfn_2.
n	The number of records or files to be copied. The default is 1.

All copying operations described in this guide begin copying from file lfn_1 at its access point and copy to lfn_2 starting at its access point. An access point is also called the current position of a file and is described in more detail later.

Copying Binary Information

The commands that copy binary information copy on a bit-by-bit basis and produce a duplicate of binary and coded data.

The following commands copy binary information.

COPYBR,lfn_1,lfn_2,n.

This command copies n records from lfn_1 to lfn_2. When the system is counting records, it increases the count by one for each EOR and by one for each EOF. If the system reaches EOI on lfn_1 before copying n records, it stops copying and adds an EOR to lfn_2. Use COPYBR when you want to copy one or more records from a specific file.

COPYBF,lfn_1,lfn_2,n.

This command copies n files from lfn_1 to lfn_2. If the system copies more than one file, the result is a multifile file. If the system reaches EOI on lfn_1 before copying n files, it stops copying and adds an EOF to lfn_2. Use COPYBF when you have a file lfn_1 that has more than one file within it (that is, when lfn_1 is a multifile file) and you want to copy one or more of the individual files.

COPYEI,lfn_1,lfn_2,x.

This command copies file lfn_1 to file lfn_2 until the system encounters an EOI on lfn_1. If the third parameter is present, the system rewinds both files before copying and then, after copying, rewinds, verifies, and rewinds both files. The third parameter (x) can be any one- to seven-character alphanumeric string. The verification is a bit-by-bit comparison of the copy with the original. Errors, if found, are identified in the job output. Use COPYEI when you want to copy one entire file to another file. For example, the command:

For example, the command:

```
COPYEI,BOX,BOTTLE,VERIFY.
```

rewinds files BOX and BOTTLE, copies file BOX to file BOTTLE, rewinds both files again, verifies that file BOTTLE is the same as file BOX, and rewinds the files a third time.

For some formats of magnetic tape files, changes may occur when copying coded data (refer to Volume 3, System Commands).

When the user enters the following sequence of copying commands, he or she creates three new files from two old files (refer to figure 4-2). In this example, the old files are nonprimary files that are positioned at BOI.

```
COPYEI,ORIG1,NEW1.
COPYBR,ORIG2,NEW2.
COPYBR,ORIG2,NEW3.
```

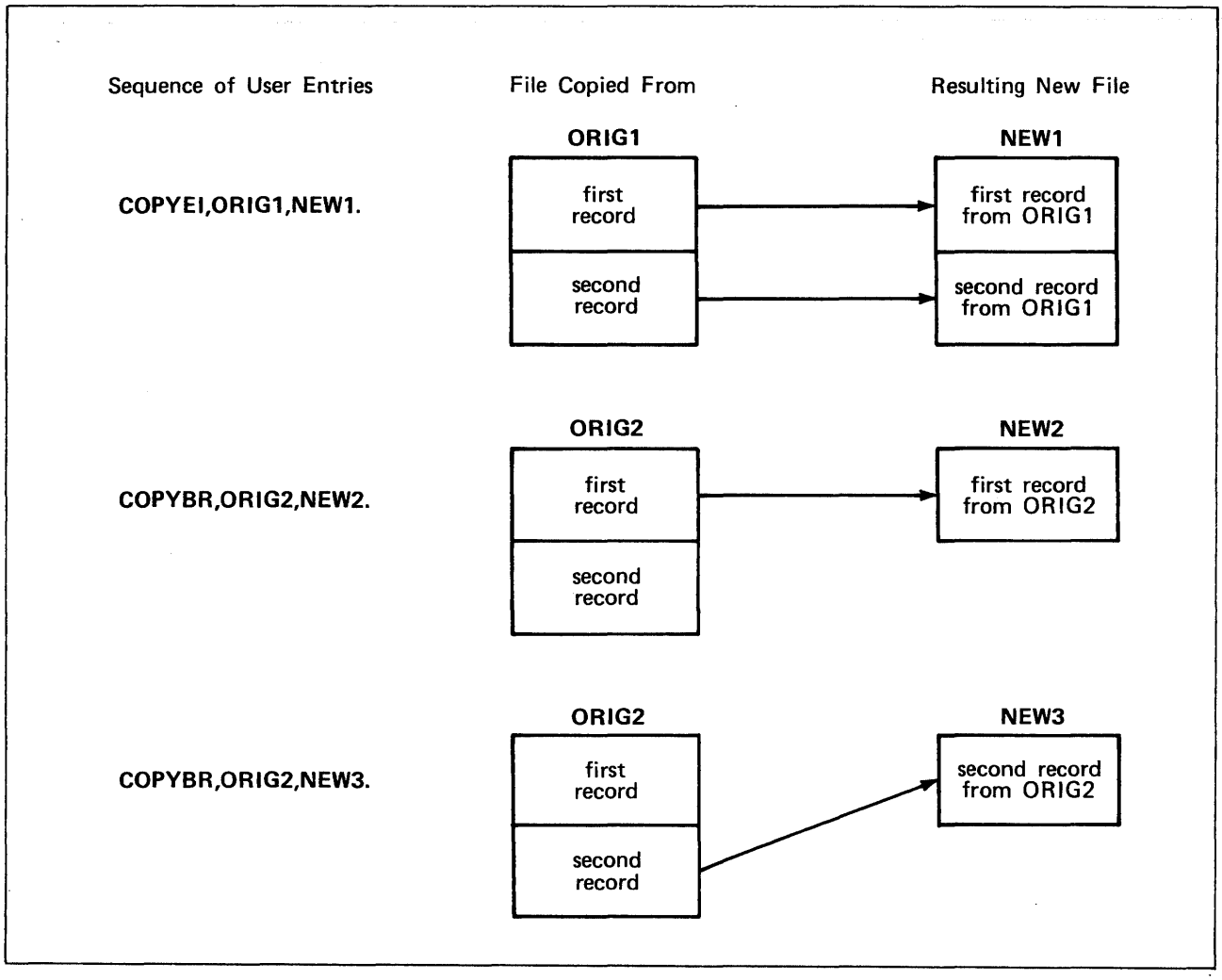


Figure 4-2. File Creation with Binary Copying Commands

Copying Coded Information

The commands that copy information in coded lines (program output, text files, and so on) copy on a character-by-character basis. Each line copied can contain a maximum of 160 characters. You would usually use such a copying command to reformat a coded file.

NOTE

We recommend that you do not use the copying commands for coded information to copy files that have both uppercase and lowercase characters: the results are unpredictable. Instead you should use the copying commands for binary information.

If you want to copy only parts of the lines of a file, use the following parameters to determine which part of each line gets copied.

Parameter	Description
fchar	The character position within each line where copying is to begin. fchar can be from 1 through 160. If you omit fchar, copying begins at character position 1.
lchar	The character position within each line where the copying is to end. lchar can be from 1 through 160 and must be greater than or equal to fchar. If you omit lchar, copying ends at character position 136.

The following commands copy coded information.

COPYCR,lfn_1,lfn_2,n,fchar,lchar.

This command copies n coded records from lfn_1 to lfn_2. If you reformat lines using fchar or lchar, the system left-justifies each line as it copies it to file lfn_2.

If the system reaches EOI or EOF on lfn_1 before copying n records, it stops copying and adds an EOR to lfn_2.

COPYCF,lfn_1,lfn_2,n,fchar,lchar.

This command copies n coded files from lfn_1 to lfn_2. If the system copies more than one file, the result is a multifile file. If you reformat lines using fchar or lchar, the system left-justifies each line as it copies it to file lfn_2.

If the system reaches EOI before copying n files, it stops copying and adds an EOF to lfn_2.

COPYSBF,lfn_1,lfn_2,n.

This command copies n coded files from lfn_1 to lfn_2, shifting each line one character position to the right and adding a leading space to all lines except the first line of each record. In the first character position of the first line of each record it places a 1.

Reformatting a file in this manner allows you to print a file on the central site line printer, which uses the first character position for carriage control. A space in the first position means single space the output; a 1 in the first position means start a new page. Refer to the discussion of line printer carriage control in Volume 3, System Commands if you want more information on printing files on a line printer.

If the system reaches EOI before copying n files, it stops copying and adds an EOF to lfn_2.

SCOPY,lfn_1,lfn_2,n,fchar,lchar.

This command copies n coded files from lfn_1 to lfn_2 and indicates the structure of the lfn_1 by displaying the EORs and EOFs in lfn_2. If the system reaches EOI before copying n files, it stops copying and adds an EOF to lfn_2.

Example:

The following list is an employee file called EMPL.

```
1445    JONES, E. T.        J22        66.25     GS7
1446    KING, L. A.         K921       49.88     GS7
1447    LANG, R. B.         L404       43.35     GS7
1448    LONG, T. M.         L11        38.97     GS11
```

The file contains the sequence number, name, employee number, deductions, and grade level for individual employees. The employee name starts in column 7; the employee number ends in column 30. To create a file NEWEMPL that contains only the employee name and number, you use the following command.

COPYCF,EMPL,NEWEMPL,1,7,30.

This command copies the characters in columns 7 through 30, left-justifying them in file NEWEMPL. The new file is:

```
JONES, E. T.        J22
KING, L. A.         K921
LANG, R. B.         L404
LONG, T. M.         L11
```

Copying Magnetic Tapes

The COPY command is often used when copying to and from tapes and so is explained in the Magnetic Tape Processing section later in this guide.

USING UPPERCASE AND LOWERCASE CHARACTERS

During normal terminal operations (normal mode), the system translates all lowercase letters you enter to uppercase.

The ASCII command allows you to interactively create files in uppercase and lowercase letters. If you use this command, you will be able to create procedure prompts and descriptions in uppercase and lowercase characters (we discuss procedures in section 8). The ASCII command's format is:

```
ASCII
```

You cannot use lowercase letters when creating jobs to be submitted in batch jobs, nor in any part of a procedure other than prompts and descriptions. When you want to return to normal mode, enter the command:

```
NORMAL
```

POSITIONING LOCAL FILES

File-positioning commands allow you to change the access point of a file, that is, the position in the file where the next operation will begin. This is also called the current position of the file. The access point of a primary file is BOI, because the system automatically rewinds the primary file to BOI before executing a command that affects it. The access point of nonprimary files is the position of the file at the end of the last operation on it. For example, assume local file DDD has two separate records and is positioned at the beginning of the first record when you enter the following commands.

```
COPYCR,DDD,EEE.
COPYCR,DDD,FFF.
```

If file DDD is the primary file, then files EEE and FFF contain the same information (the first record of file DDD). This is because the system rewinds the primary file before copying it. If file DDD is not the primary file, file EEE contains the first record of file DDD, and file FFF contains the second record (refer to figure 4-3).

You can move the access point of a file forward (toward EOI) using one of the following commands.

Command	Description
SKIPR	Move forward by records.
SKIPF	Move forward by files.
SKIPEI	Move to EOI.

You can move the access point of a file backward using one of the following commands.

Command	Description
BKSP	Move backward by records.
SKIPFB	Move backward by files.
REWIND	Move to BOI.

In the following discussion of these commands, only the principal parameters are explained; all parameters for each command are given in Volume 3, System Commands.

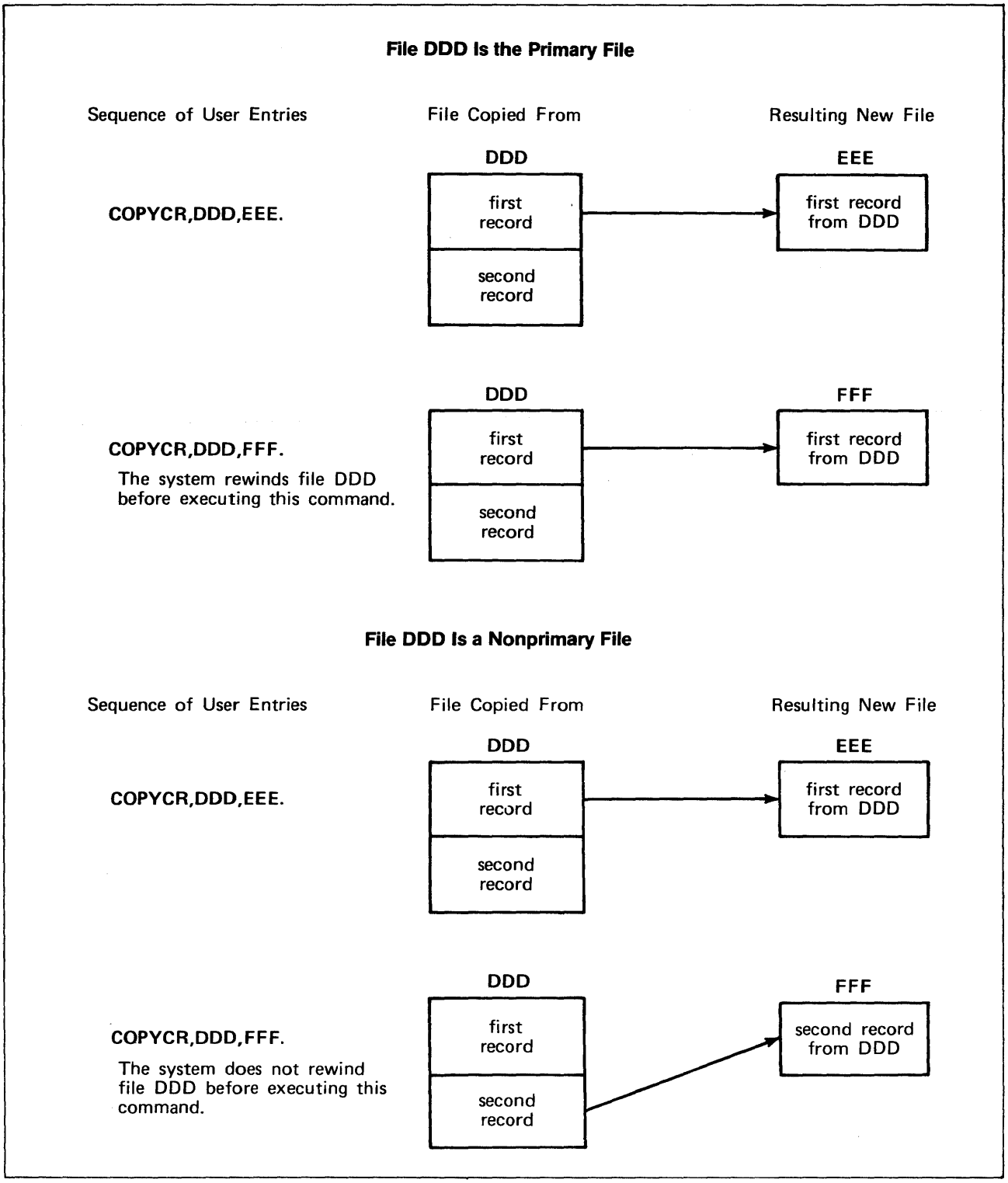


Figure 4-3. Copying Coded Records

SKIPR COMMAND

The SKIPR command moves the access point of the file forward (toward EOI) by records. Its format is:

SKIPR,lfn,n.

In this command, lfn is the name of the local file, and n is the number of records skipped. An EOF is considered a separate record and is included in the record count. If the system reaches EOI before skipping n records, the access point is at EOI.

Figure 4-4 shows three examples of the SKIPR command.

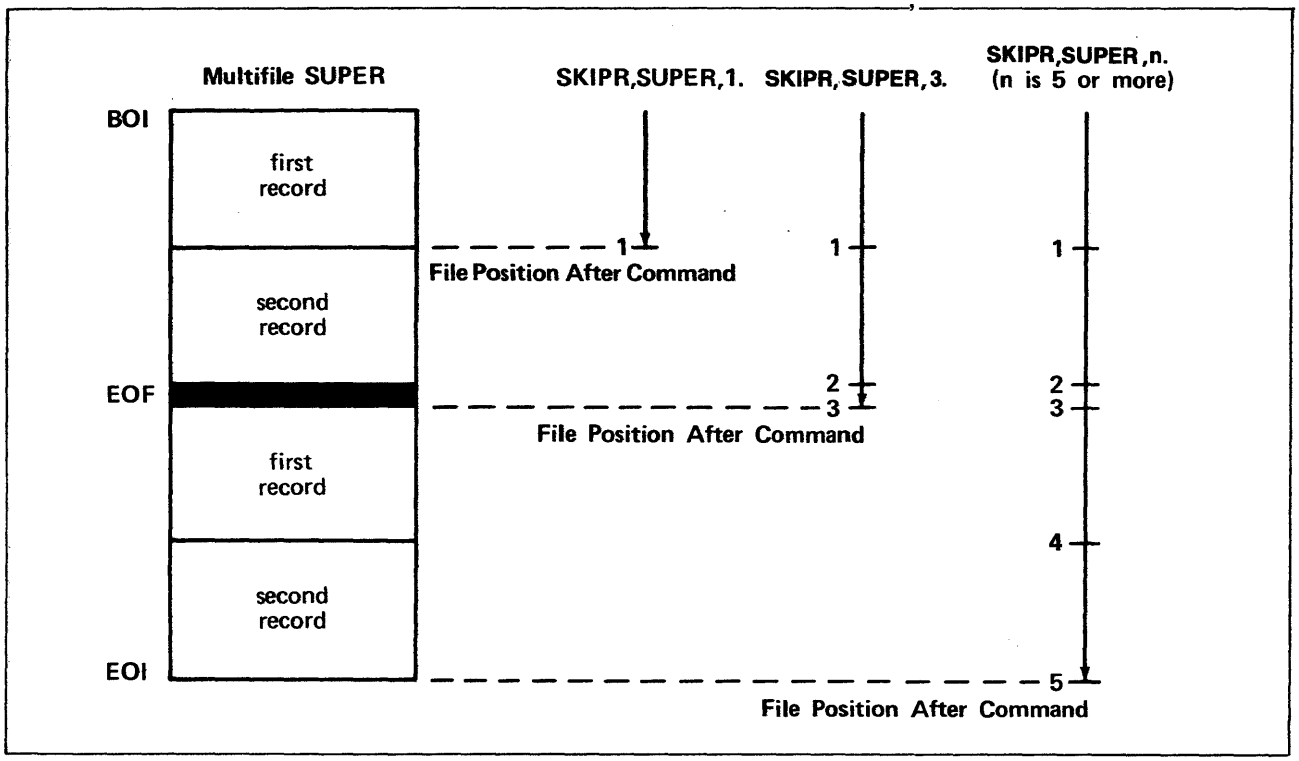


Figure 4-4. SKIPR Command

SKIPF COMMAND

The SKIPF command moves the access point of a multifile file forward (toward EOI) by files. Its format is:

SKIPF,lfn,n.

In this command, lfn is the name of the local file, and n is the number of files skipped. If the system reaches EOI before skipping n files, the access point is at EOI.

SKIPEI COMMAND

The SKIPEI command moves the access point of a file to EOI. On magnetic tapes where no EOI is defined the file position is set at the next EOF on the file. The command's format is:

SKIPEI,lfn.

In this command, lfn is the name of the file to be positioned.

BKSP COMMAND

The BKSP command moves the access point backward (toward BOI) by records. Its format is:

BKSP,lfn,n.

In this command, lfn is the name of the local file, and n is the number of records skipped. An EOF is considered a separate record and is included in the record count. If the system reaches BOI before skipping n records, the access point is at BOI.

Figure 4-5 shows the use of the BKSP command.

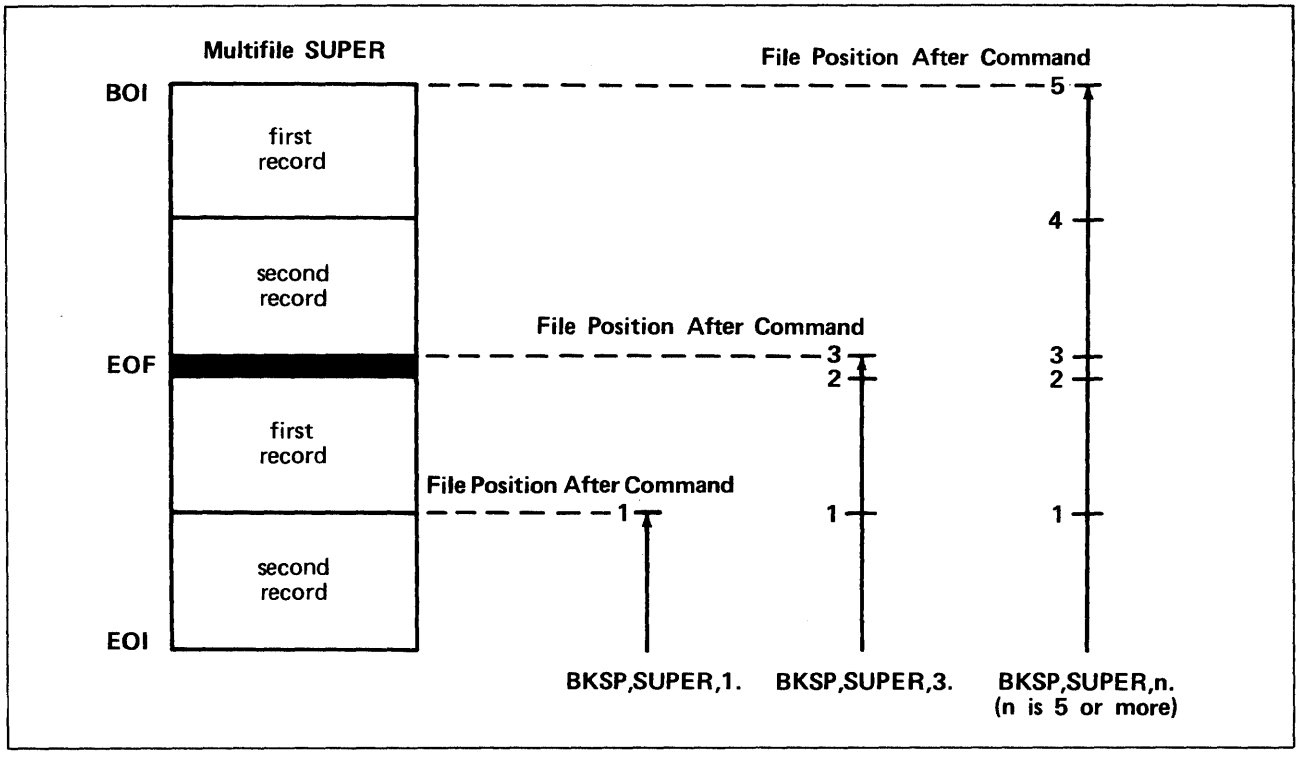


Figure 4-5. BKSP Command

SKIPFB COMMAND

The SKIPFB command moves the access point backward (toward BOI) by files. Its format is:

SKIPFB,lfn,n.

In this command, lfn is the name of the local file, and n is the number of files skipped. If the system reaches BOI before skipping n files, the access point is at BOI.

REWIND COMMAND

The REWIND command positions mass storage files at BOI (rewinding tape files is discussed in Magnetic Tape Processing, section 6).

Many operations on nonprimary files leave files at EOI. A file may be at EOI if the system's only response to a command you enter is:

EOI ENCOUNTERED.

The REWIND command's formats are:

REWIND,lfn$_1$,lfn2,...,lfn$_n$.

This command rewinds local files lfn$_1$,...,lfn$_n$.

REWIND,*.

This command rewinds all local files.

REWIND,*,lfn$_1$,lfn2,...,lfn$_n$.

This command rewinds all local files except files lfn$_1$,...,lfn$_n$.

CHANGING NAMES OF LOCAL FILES

You can use the RENAME command to change the name of a local file. Its format is:

```
RENAME,nlfn=olfn.
```

In this command, nlfn is the new file name you want the local file called, and olfn is the old file name previously associated with the local file.

If another local file is called nlfn, the system releases it when it changes file olfn to file nlfn. You cannot use the RENAME command to change the name of a permanent file. If a local file is associated with a permanent file and you change the name of the local file, the permanent file name does not change.

RELEASING LOCAL FILES

Releasing a local file means that the file is no longer associated with your job. If the file being released is a direct access file, the file remains as a permanent file. Otherwise, the information in the file is not retained unless you make it permanent before you release the local file.

You can release local files by:

- Creating a new primary file with the NEW or OLD command.
- Using the RETURN or UNLOAD command (explained later).
- Completing a job. When you log out or when the system has finished processing your batch job, all your local files are released.

Local files can have a special no-auto-drop status that prevents them from being released from your job unless you name them specifically on the RETURN or UNLOAD command. Some files created and used by the system automatically have this status. The files used in this section do not have no-auto-drop status. If you are interested in using it, you can specify the status with the SETFS command (refer to Volume 3, System Commands).

Figures 4-6 and 4-7 show different ways of releasing files, most of which you are familiar with. We discuss the remaining method, the RETURN command, next.

RETURN COMMAND

The RETURN command releases local files. Its formats are:

RETURN,lfn_1,lfn_2,...,lfn_n.

This command releases local files lfn_1,...,lfn_n even if they have no-auto-drop status.

RETURN,*.

This command releases all local files that do not have the no-auto-drop status.

RETURN,*,lfn_1,lfn_2,...,lfn_n.

This command releases all local files that do not have the no-auto-drop status except files lfn_1,...,lfn_n.

If you are using more than one tape or removable disk at the same time, your system resources may decrease when you release a file with the RETURN command.

Sequence of User Entries	Local Files		Comments
	Primary	Nonprimary	
OLD,DESK.	DESK		Makes a local copy of permanent file DESK and designates it the primary file; releases all other files.
GET,VARNISH.	DESK	VARNISH	Makes a local copy of file VARNISH; does not release other local files.
FTN5,I=DESK,L=WORK,B=BRUSH,GO.	DESK	VARNISH WORK BRUSH	Causes local files WORK and BRUSH to be created by the FORTRAN compiler; does not release other local files.
RETURN,VARNISH.	DESK	WORK BRUSH	Releases local file VARNISH; does not release other local files.
NEW,HOBBY.	HOBBY		Creates a new local file named HOBBY and designates it the primary file; releases all other local files.
BYE.			Logs out; releases all local files.

Figure 4-6. Releasing Local Files

Sequence of User Entries	Local Files: Primary	Local Files: Nonprimary	Comments
NEW,PARTY.	PARTY		Creates an empty primary file named PARTY; releases all other local files.
OLD,DANCE/ND.	DANCE	PARTY	Makes a copy of permanent file DANCE to be the primary file; does not release other files, and file PARTY becomes a nonprimary file.
NEW,PARTNER/ND.	PARTNER	PARTY DANCE	Creates an empty primary file named PARTNER; does not release other files, and file DANCE becomes a nonprimary file.
RETURN,*,PARTY.		PARTY	Releases all files except file PARTY.
OLD,WORK=JOB.	WORK		Makes a copy of file JOB, names the copy WORK, and makes WORK the primary file. All other local files are released.

Figure 4-7. Releasing Local Files Using Special Parameters

UNLOAD COMMAND

The UNLOAD command releases local files. Its formats are:

UNLOAD,lfn_1,lfn_2,...,lfn_n.

This command releases local files lfn_1,...,lfn_n even if they have no-auto-drop status.

UNLOAD,*.

This command releases all local files that do not have the no-auto-drop status.

UNLOAD,*,lfn_1,lfn_2,...,lfn_n.

This command releases all local files that do not have the no-auto-drop status except files lfn_1,...,lfn_n.

The system does not change the resource demand count established by the RESOURC command as it may if you use the RETURN command (section 6 has more information).

PERMANENT FILES 5

Permanent files are files that are stored in the system so that you can keep them from job to job. These files remain in the system until you or someone at your site removes them.

All permanent files are classified according to the manner in which they are accessed: indirect or direct access. The mode of access is determined by the command used to make the file permanent. How the file is made permanent also determines the command you must enter to access the file. The characteristics for each type of file follow.

Access Mode	Characteristics
Indirect	You can make a temporary file permanent and classify it as an indirect access file by using either the REPLACE or SAVE command (described later). To access such a permanent file, you use the OLD or GET command. When you enter the GET or OLD command, the system makes a copy of the permanent file. The system uses the copy, not the permanent file, in all subsequent operations. This feature allows you to make changes to a copy of a file without changing the permanent file. However, since only the copy of the file is changed, if you do want to change the permanent file, you must replace it with your changed copy using the REPLACE command. If you want to keep both your changed copy and the unchanged permanent file, you can create a new permanent file using the changed copy. You do this by specifying a different permanent file name on the SAVE command. The changed copy becomes a new indirect access permanent file. The system allocates mass storage for indirect access permanent files in blocks of one or more PRUs. Each PRU contains 64 words. The block size allocated for direct access files is much larger than one PRU. Thus, to avoid using more mass storage space and system resources than are necessary, your small files should be indirect access permanent files.
Direct	You can create a file that is both local and permanent by using the DEFINE command. Such a file is called a direct access file because, when you subsequently access the file, the system associates the actual permanent file with your job rather than making a copy. Using the ATTACH command gains access to an existing direct access file. Direct access files have a greater input/output efficiency than indirect access files because no intermediate copy is used. However, you should take care in changing a direct access file, since all changes (correct and erroneous) are made directly to the permanent file, not a copy. The system allocates mass storage for direct access permanent files in large blocks; the block size depends on the type of mass storage device on which the file resides (for more information read Mass Storage File Residence in Volume 3, System Commands). Therefore, direct access files are more efficient for large files such as data base files or user libraries.

The user name you specify either during login or on the USER command is considered the owner of all files made permanent during that job. Normally, you have access only to those files that were made permanent under your user name. However, you can access files belonging to a different user, if that user allows you access (the procedure is described later in this section).

CREATING AN INDIRECT ACCESS FILE

You can create indirect access permanent files from temporary files with the SAVE and REPLACE commands. These commands make a copy of your file for permanent storage, and also retain the file for use in your job.

The SAVE command's format is:

SAVE,lfn=pfn.

Option	Description
lfn	The local file name of the file that you want saved.
pfn	The name you want to give to the permanent file you are creating. If a permanent file already exists with this name, the system returns the message **pfn ALREADY PERMANENT.** and does not create a new permanent file with the same name. You must either change pfn, use the REPLACE command, or change the name of or purge the existing permanent file (the CHANGE and PURGE commands are described later). If you specify only **SAVE,lfn.** the local file name also becomes the name of the permanent file.

The REPLACE command replaces an existing indirect access file with a new permanent file. Its format is:

REPLACE,lfn=pfn.

Option	Description
lfn	The local file name of the file that you want saved.
pfn	The name you want to give to the permanent file. If a permanent file already exists with this name under your user name, the system writes over the existing information in the file. If you specify only **REPLACE,lfn.** the local file name also becomes the name of the permanent file.

If you name no files on the SAVE and REPLACE commands, a copy of the primary file becomes an indirect access file.

ACCESSING AN INDIRECT ACCESS FILE

You can create a local copy of an indirect access file with the GET and OLD commands. Any changes you make to the local copy are not made on the permanent file. If your changes prove to be worthwhile, you can then either save a copy of the local file as a new permanent file or replace the existing permanent file with a copy of the local file.

The example in figure 5-1 shows how to create and access indirect access permanent files during a terminal session.

Terminal Session	Description
`/get,pik=piklow`	The user makes a copy of permanent file PIKLOW and calls it PIK.

`/modify,p=pik,n=piklow,f`
`.`
`.`
`.`
`MODIFICATION COMPLETE.`
	The user modifies file PIK using the Modify utility. Modify creates new local file PIKLOW, which contains the new changes. Modify is discussed in section 9.

`/replace,piklow`	The user replaces permanent file PIKLOW with newly created local file PIKLOW.
`/old,shoe`	The user makes a copy of permanent file SHOE and designates it the primary file.

`/fse,shoe`
`CREATE: SHOE`
`NOS FULL SCREEN EDITOR`
`??`
`.`
`.`
`.`
`?? quit`
`FILE: SHOE (NO CHANGES)`
`FILE: FSEPROC (NO CHANGES)`
	The user modifies file SHOE using FSE, a text editor. Examples using FSE are given and explained in later sections.

`/save,shoe=boot`	The user creates a new indirect access file named BOOT. BOOT contains the same information that is on local file SHOE. Permanent file SHOE remains unchanged.

Figure 5-1. Accessing and Creating Indirect Access Files

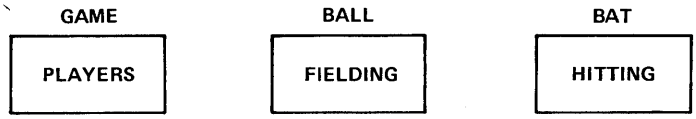
ADDING INFORMATION TO AN INDIRECT ACCESS FILE

You can add information to the end of an indirect access file with the APPEND command. This information must be in one or more local files. The system adds a copy of the entire file or files, including EORs and EOFs, to the permanent file. The APPEND command's format is:

APPEND,pfn,lfn_1,...,lfn_n.

Option	Description
pfn	The name of the indirect access permanent file to which local files will be added.
lfn_i	The names of the local files you want added to the permanent file. They are added in the order in which they are listed.

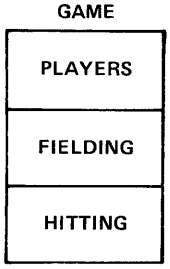
For example, file GAME (an indirect access permanent file) and files BALL and BAT (both local files) have one record each, as follows:



If you enter:

APPEND,GAME,BALL,BAT.

file GAME becomes:



The system adds files BALL and BAT to the end of permanent file GAME, separating them by EORs. The APPEND command affects only permanent files. If there also is a local file named GAME, the local file is unchanged.

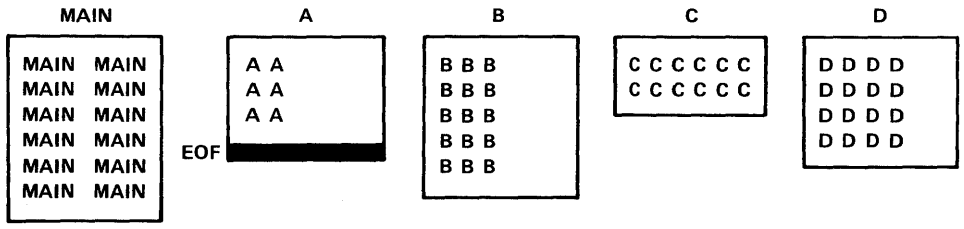
The system stops processing the APPEND command if it reaches a file that is to be appended that is not a local file. For example, suppose that files A, B, and D are local files and you enter:

APPEND,MAIN,A,B,C,D.

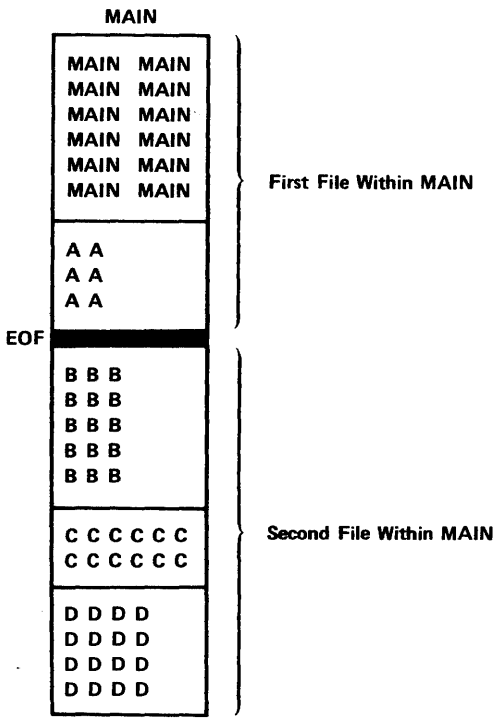
The system will append files A and B to file MAIN, and then stop processing the command because file C is not a local file. If this happens, you get file C and enter the APPEND command as follows:

```
GET,C.
APPEND,MAIN,C,D.
```

File MAIN now contains the information that was originally on files MAIN, A, B, C, and D. The structure of the file is dependent upon the structure of the original files. If any of the original files contain an EOF, the file distinction remains and the resulting file becomes a multifile file. Suppose files MAIN, A, B, C, and D have one record each, and file A also has an EOF:



Permanent file MAIN, after appending files A, B, C, and D, becomes a multifile file which looks like:



If you also have a local file named MAIN, the local file remains unchanged.

MODIFYING AN INDIRECT ACCESS FILE

If you want to modify one of your indirect access files, you can:

- Create a new local file and replace the existing permanent file with the new local file.
- Get a local copy of the permanent file, make changes to the local copy, and then replace the existing permanent file with the changed local file. There are many different ways to change local copies. Section 9 discusses using a text editor or a source code maintenance utility, and section 4 describes commands that you can use.

No matter which method you choose, the final step is to replace the permanent file with its changed version. You can do this with the REPLACE command. Its format is:

```
REPLACE,lfn=pfn.
```

Option	Description
lfn	The name of the local file that contains the updated information.
pfn	The name of the permanent file that is to be replaced.

If the local file and the permanent file have the same name, you can enter

```
REPLACE,pfn.
```

Example:

A manufacturing firm maintains a list of parts on an indirect access file named PARTS. Within this file, the parts for each product are located on a separate record. There are 18 records in the file. The firm introduces two new products and cancels one product. The parts for the new products are located on indirect access files NEW1 and NEW2. The parts for the canceled product are on record 13. Figure 5-2 shows an interactive session that modifies file PARTS.

Terminal Session	Description
/get,new1 /get,new2 /get,parts	The user makes local copies of the two new parts files and the complete parts file.
/copybr,parts,list,5 COPY COMPLETE.	The user copies the first five records of file PARTS to file LIST.
/copybr,new1,list,1 COPY COMPLETE.	The user copies the first record on file NEW1 to file LIST.
/copybr,parts,list,7 COPY COMPLETE.	The user copies the next seven records of file PARTS to file LIST.
/skipr,parts,1 SKIPR,PARTS,1.	The user skips the next record (record 13) in file PARTS.
/copybr,parts,list,3 COPY COMPLETE.	The user copies the next three records of file PARTS to file LIST.
/copybr,new2,list,1 COPY COMPLETE.	The user copies the first record of file NEW2 to file LIST.
/copybr,parts,list,2 COPY COMPLETE.	The user copies the last two records of file PARTS to file LIST.
/replace,list=parts	The user replaces indirect access file PARTS with its modified version, file LIST.

Figure 5-2. Modifying an Indirect Access File

CREATING A DIRECT ACCESS FILE

Using the DEFINE command, you can create a new, empty, local file and simultaneously classify it as a direct access permanent file. When you access the permanent file in subsequent jobs, the system associates the actual permanent file with your job, rather than a copy of the file. A file created with the DEFINE command is a nonprimary file.

```
DEFINE,lfn=pfn.
```

Option	Description
lfn	The local file name the system gives the file while it is associated with your job.
pfn	The permanent file name.

You can use the lfn=pfn parameter if you want to reference the file by a name other than its permanent file name. If you specify only

```
DEFINE,pfn.
```

the system assumes that the local file name and permanent file name are to be the same.

The following example creates an empty direct access file named MIDDLE and copies local files LOW and HIGH to file MIDDLE. File MIDDLE is then rewound and printed at the terminal.

Terminal Session	Description
/define,middle	The user creates a direct access file.
/get,low /get,high	The user makes local copies of indirect access files LOW and HIGH.
/copybf,low,middle EOI ENCOUNTERED. /copybf,high,middle EOI ENCOUNTERED. /rewind,middle REWIND,MIDDLE.	The user copies files LOW and HIGH to file MIDDLE.
/copy,middle VALLEY ZERO GRADE SCHOOL LABOR MOUNTAIN 1 BILLION COLLEGE MANAGEMENT EOI ENCOUNTERED.	The user prints file MIDDLE at the terminal.

ACCESSING A DIRECT ACCESS FILE

You can associate a direct access file with your interactive or batch job by using the ATTACH command. You access the permanent file itself, not a copy of the permanent file. The ATTACH command's format is:

```
ATTACH,lfn=pfn.
```

Option	Description
lfn	The local file name the system gives the direct access file while it is associated with your job.
pfn	The permanent file name of the direct access file.

If you specify only:

```
ATTACH,pfn.
```

the system uses the permanent file name as the local file name.

MODIFYING A DIRECT ACCESS FILE

If you want to modify a direct access file, you must include the M=W parameter at the time that you attach the file. The command's format is:

```
ATTACH,lfn=pfn/M=W.
```

Parameter	Description
M=W	Specifies write mode and indicates that you want to be able to change the file.

If you do not include /M=W on the command, the system will not allow you to make changes to the file. This is a precaution against inadvertently modifying a direct access file.

After you attach the direct access file in write mode, you can use the copy and file positioning commands to alter the file. Many of these commands are discussed in section 4. The following example shows a record being added to the end of a direct access file named LISTING. The new record is on file ADD.

Terminal Session	Description
/get,add	The user makes a copy of file ADD.
/attach,listing/m=w	The user attaches file LISTING in write mode.
/skipei,listing SKIPEI,LISTING.	The user positions file LISTING at EOI.
/copybr,add,listing,1 COPY COMPLETE.	The user copies one record from file ADD to the end of file LISTING.

MANIPULATING FILES

As you use files during a job, you may want to be able to access more than one file at a time or to make changes to a file and then use that file later in the job. The commands you use to access the files determine whether files you have previously accessed are released. Keep in mind that:

- You can have only one local file with a given name. This name may or may not be the same as the name of one of your permanent files.
- You can have only one primary file. Primary files are useful because in many commands the default file name is the name of the primary file. Also, the system always positions them at the beginning of the file before executing a command.
- Accessing a file as a nonprimary file does not release any files, unless a file has the same local file name you used with the GET or ATTACH command.
- Using the SAVE or REPLACE command does not release the local file you name. It does position it to the beginning of the file.
- Changes made to local direct access files are permanent; changes made to other local files are not permanent, unless you save the changes.

The example in figure 5-3 shows commands that manipulate local and permanent files.

REMOVING PERMANENT FILES FROM THE SYSTEM

You can remove your indirect and direct access permanent files from the system by entering the PURGE command. Its basic format is:

PURGE,pfn_1,pfn_2,...,pfn_n.

Parameter	Description
pfn_i	The names of the permanent files, created under your user name, that are to be removed.

The PURGE command does not affect local files. For example, if you purge a direct access file while it is a local file (associated with your job), it remains available to you. However, when you release it, it disappears completely.

Your site may have a policy of purging permanent files that have not been used for a specified length of time. Check with site personnel for information.

Terminal Session	Description
. . .	
/old,project	The user makes file PROJECT the primary file. All other local files are released.
/get,acct1 /attach,list3/m=w	The user makes files ACCT1 and LIST3 local files. File PROJECT remains the primary file.
. . .	
/return,list3,acct1 RETURN,LIST3,ACCT1.	The user releases files LIST3 and ACCT1. The user did not make local file ACCT1 a permanent file, and therefore, any changes made are not kept. LIST3 is a direct access file and therefore, as changes were made to file LIST3, they became part of the permanent file.
/get,acct4 /attach,list4/m=w	The user makes files ACCT4 and LIST4 local files. File PROJECT remains the primary file.
. . .	
/replace,acct4	The user replaces permanent file ACCT4 with a copy of local file ACCT4: changes are kept.
. . .	
/new,fun	The user creates empty file FUN, releasing all other files, including files PROJECT, ACCT4, and LIST4. Any changes made to file PROJECT and any additional changes to file ACCT4 are not kept. Changes made to file LIST4 are kept.

Figure 5-3. Manipulating Local and Permanent Files

GRANTING OTHER USERS ACCESS TO FILES

When you create a permanent file you can grant other users permission to access it. To do this, you add the file permit category parameter to the SAVE or DEFINE command. The parameter's format is:

CT=ct

ct	Description
P	Private file. If you specify CT=P, no one other than yourself can access the file unless you specifically give him or her permission with the PERMIT command (discussed later in this section). Private is the default permit category.
S	Semiprivate file. If you specify CT=S, all users who know the file name, your user name, and the password (described later) can access the file. The system keeps a list of individual accesses to the file by other users. You can access this information with the CATLIST command (explained later).
PU	Public file. If you specify CT=PU, the access requirements for other users are the same as for the semiprivate file category. However, the system keeps track only of the total number of accesses by other users.

In addition to establishing a permit category to determine who can access the file, you can also establish an access mode to determine how another user can access the file. You specify this mode on the SAVE or DEFINE command with the following parameter.

M=m

m	Description
E	Execute. If you specify M=E, other users can only execute the file.
R	Read. If you specify M=R, other users can read, execute, and make their own copies of the file. The default mode for both indirect and direct access files is read.
A	Append. If you specify M=A, other users can add information to the end of the file and make their own copies of the file.
W	Write. If you specify M=W, other users can alter, execute, and read the file; add information to the end of the file; make their own copies of the file; and purge the permanent file.

The restrictions you place on another user's use of your indirect access file do not apply to their local copy of that file. However, they cannot change your permanent file unless you specify either A or W as their access mode. The restrictions you place on another user's use of your direct access file apply while the file is associated with their job.

Other access modes are also available. Refer to Volume 3, System Commands for a complete list of access modes.

How you limit the access of your permanent files by others does not affect your ability to access them. When you access an indirect access file that you have created, you automatically have write permission even though others may not be able to alter the permanent file. For a direct access file, the default access mode for all users is read. If you want to alter one of your direct access files, you must specify M=W when you attach it.

Users who have permission to access one of your files cannot obtain CATLIST information on the file unless you have explicitly specified that the file can be listed by alternate users. You specify this permission on the SAVE or DEFINE command with the following parameter.

AC=ac

ac	Description
Y	YES. Users who have permission to access the file can display CATLIST information on the file.
N	NO. Other users cannot display CATLIST information on the file. NO is the default value.

When you create a permanent file, you also can associate a password with the file. If you do, all other users must specify the password, along with the file name and your user name, to access the file. The password for the file has no relationship to the password you specify when you log in or use the USER command. To associate a password with a file, you specify the following parameter on a SAVE or DEFINE command.

PW=password

Option	Description
password	A 1- to 7-character alphanumeric password.

INDIRECT ACCESS FILES

To create an indirect access file that can be accessed by others, you use the following format of the SAVE command.

```
SAVE,lfn=pfn/PW=password,CT=ct,M=m.
```

Another user who wants to access the indirect access file uses the following format of the GET command.

```
GET,lfn=pfn/UN=username,PW=password.
```

Option	Description
username	The user name assigned to the person who created the file.

If you do not specify a password when you create the file, another user does not use the PW=password parameter when accessing the file.

The PW=password, CT=ct, M=m, and UN=username parameters can appear in any order after the slant. The slant is a required separator that must come after the file names, if you use any of the parameters granting access to other users.

Example:

With the following command, a user with user name RST441 creates indirect access file ALT from local file LFILE and allows other users to read the file.

```
SAVE,LFILE=ALT/PW=PASS,CT=S,M=R.
```

To access file ALT during subsequent jobs, user RST441 enters:

```
GET,ALT.
```

To access file ALT, another user enters:

```
GET,ALT/UN=RST441,PW=PASS.
```

DIRECT ACCESS FILES

To create a direct access file that can be accessed by others, you use the following format of the DEFINE command.

```
DEFINE,lfn=pfn/PW=password,CT=ct,M=m.
```

Another user who wants to access the direct access file uses the following format of the ATTACH command.

```
ATTACH,lfn=pfn/M=m,UN=username,PW=password.
```

Option	Description
username	The user name assigned to the person who created the file.

If you specify that another user can access the file in other than read mode, the other user must include the M=m parameter on the ATTACH command to access the file in other than read mode. If the other user specifies a mode that would allow her or him to do more to the file than you specified, the other user is not allowed to access the file. For example, if you specify M=R on the DEFINE command, and the other user specifies M=W, the system does not allow access. However, if you specify M=W and the user specifies M=R, the system allows the user to read and/or execute the file.

As with indirect access files, if you do not specify a password when you create a direct access file, another user does not specify one when accessing the file.

The PW=password, CT=ct, M=m, and UN=username parameters can appear in any order after the slant. The slant is a required separator that must come after the file names, if you use any of the parameters granting access to other users.

Example:

A user with user name AA42 wants a file to have these characteristics:

- Its local file name is ICE.
- Its permanent file name is COLD.
- Its password is DRINK.
- Its permit category is SEMIPRIVATE.
- Alternate users can access it in WRITE mode.
- Alternate users can list file information using CATLIST.

The DEFINE command that creates a direct access file with these characteristics is:

```
DEFINE,ICE=COLD/PW=DRINK,CT=S,M=W.
```

To access file ICE during subsequent jobs, user AA42 enters either:

```
ATTACH,COLD. or ATTACH,COLD/M=W.
```

The first entry is for read mode; the second entry is for write mode.

To access the file in write mode, another user enters:

```
ATTACH,COLD/M=W,UN=AA42,PW=DRINK.
```

SPECIFIC USERS

With the PERMIT command, you can allow specific users to access a private file, specifying a particular mode of access for each user. You can also rescind the access permission you granted in a previous PERMIT command and, for semiprivate files, deny access to specific users. The PERMIT command's format is:

PERMIT,pfn,username$_1$=m$_1$,username$_2$=m$_2$,...,username$_n$=m$_n$.

Option	Description
pfn	The name of the permanent file.
username$_i$	The user names of those who are being granted access to the private or semiprivate file.
m$_i$	Access modes. Some of the possible modes are:

m$_i$	Description
E	Execute.
R	Read, execute, and copy (default mode).
A	Append and copy.
W	Write, append, read, execute, copy, and purge.

Complete descriptions of access modes are in Granting Other Users Access to Files earlier in this section.

Example:

If you have created a private file named CUP and want to grant access to:

User Name	Mode of Access
RFK12	Write
STP45	Read

you enter:

```
PERMIT,CUP,RFK12=W,STP45=R.
```

If you do not want a specific user to access a semiprivate file, or you want to remove permissions you gave to a user for a private file, you can enter NULL as the access mode on the PERMIT command.

Example:

If file ABC is a semiprivate file and you do not want a user with user name DUL684 to access the file, you enter:

```
PERMIT,ABC,DUL684=NULL.
```

CHANGING PERMANENT FILE CHARACTERISTICS

You can change many of the characteristics of permanent files you have created. The CHANGE command performs this function, and its format is:

```
CHANGE,nfn=ofn/PW=password,CT=ct,M=m,AC=ac.
```

Parameter	Description
nfn=ofn	The new file name (nfn) that replaces the old file name (ofn). If you do not want to change the permanent file name, specify only ofn.
PW=password	The new 1- to 7-character password you want associated with the file. Use PW=password when you want either to add a password to a file that did not have one or to change an existing password. To remove a file's password, enter PW=0.
CT=ct	The new permit category for the file. The following entries are possible.

ct	Description
P	Private file (default category).
S	Semiprivate file.
PU	Public file.

Complete descriptions of permit categories are in Granting Other Users Access to Files in this section.

M=m — The new access mode for other users. Some of the possible modes are:

m	Description
E	Execute.
R	Read, execute, and copy (default mode).
A	Append and copy.
W	Write, append, read, execute, copy, and purge.

Complete descriptions of access modes are in Granting Other Users Access to Files in this section.

AC=ac — The new CATLIST display status for alternate users. Possible values are:

ac	Description
Y	YES. Alternate users can obtain CATLIST information.
N	NO. Alternate users cannot obtain CATLIST information.

OBTAINING PERMANENT FILE INFORMATION

Using the CATLIST command, you can obtain information about files in your permanent file catalog. You can also obtain information about files in other users' catalogs. To get information on another user's files, the file owner must have given you access permission to those files and must have made each file available to CATLIST inquiries by alternate users.

The following options are discussed in this guide.

- You can obtain a list of the names of your permanent files and the names of other user's files that you have permission to access and display.
- You can obtain general information about each file, such as its creation date and the date it was last accessed.
- You can obtain a list of other users who have permission to access a private file and their permitted mode of access.
- You can obtain a list other users who have accessed a semiprivate file and their permitted mode of access.

The information returned by the CATLIST command is called a catalog of your files.

LISTING PERMANENT FILE NAMES

To obtain an alphabetical list of the names of the permanent files created under your user name, you enter:

```
CATLIST.
```

The system first lists the indirect access files and then the direct access files and also gives the total number of each type of file. Figure 5-4 shows an example of this short list.

If you want a list of the files you can access that were created under a user name different from yours, you enter:

```
CATLIST,UN=username.
```

username is the name under which the files were created.

This form of the CATLIST command lists all files in the specified catalog which you have permission to use and for which the file owner has given alternate CATLIST display permission. Access and display permissions for a specific file are defined on a SAVE, DEFINE, or CHANGE command that names that file.

```
CATALOG OF  ABC4321          FM/NOSHOP  yy/mm/dd. 13.55.34

INDIRECT ACCESS FILES

 APPLE     BATRT     DAY1      EXAMPLE   NEWDAY    TAPFIL    T2
 BATEX     BOX       ERROREX   FILEDAY   PRESENT   T1        VDAYS
 BATEX2    DAY       ESTATE

DIRECT ACCESS FILES

 MAY

      17 INDIRECT ACCESS FILES,   TOTAL PRUS =       25.
       1 DIRECT ACCESS FILE,      TOTAL PRUS =        1.
```

Figure 5-4. Listing Your File Names

LISTING GENERAL PERMANENT FILE INFORMATION

To obtain general information about your files, you enter:

```
CATLIST,LO=F.
```

The system prints out three lines of information on each of your files. To obtain this information for only one file, you enter:

```
CATLIST,LO=F,FN=filename.
```

filename is the name of the file for which you want information.

If you want general information about another user's files that you have permission to access, you can include the UN=username parameter on either of the two preceding CATLIST commands. To list information on another user's files, however, the owner of those files must explicitly allow listing of each file, using the AC parameter on a SAVE, DEFINE, or CHANGE command.

Figure 5-5 shows a CATLIST,LO=F listing made by user FY635KG. Descriptions of the fields follow.

Field	Description
FILE NAME	The permanent file name.
FILE TYPE	The file permit category and access mode. The possible entries for the permit category are PRIVATE, SEMI-PR (for semiprivate), and PUBLIC. The possible entries for the access mode are DIR for direct access and IND for indirect access.
LENGTH	The length of the file in decimal PRUs.
DN	The device number for direct access files. An asterisk in this field indicates the file is on your master device. This device holds your catalog and all your indirect access files.

Field	Description
CREATION DATE/TIME	The creation date and time in the following format. yy/mm/dd. hh.mm.ss.
ACCESS DATE/TIME	The date and time of the last access of the file.
DATA MOD DATE/TIME	The date and time of the last modification of the file.
PASSWORD	The file password if the file has one. The password is not listed if the file belongs to another user.
COUNT	The number of times the file has been accessed.
INDEX	Reserved for system utilities.
PERM.	The permitted access mode. The entries described in this guide are EXECUTE, READ, APPEND, and WRITE.
SUBSYS	The subsystem association. If a file was saved in an interactive session, it could be associated with one of the following subsystems. Subsystem / Entry BASIC / BASIC Batch / BATCH Execute / EXEC FORTRAN (FORTRAN 5) / FORT FTNTS (FORTRAN 4) / FTNTS If this field is blank, no subsystem is associated with the file.
EXPIRES	The date on which the password expires.
LEVEL	The access level of the file.
PR	The preferred residence. These fields are used when your site has a Mass Storage Subsystem. Refer to the Volume 3, System Commands for more information.
BR	The backup requirement.
RS	The actual residence.
AC	Indicates whether alternate users can obtain CATLIST information on the file. Y indicates that file information can be listed; N indicates that it cannot. N is the default.
CHARGE NO.	The charge number associated with this file. Default is the charge number in effect at the time the file was created.
PROJECT NUMBER	The project number associated with this file. Default is the project number in effect at the time the file was created.

```
CATALOG OF  FY635KG          FM/NOSHOP 84/05/19. 12.13.36.  PAGE  1

   FILE NAME  FILE TYPE       LENGTH DN CREATION  ACCESS    DATA MOD
    PASSWORD  COUNT INDEX  PERM. SUBSYS DATE/TIME DATE/TIME DATE/TIME
     EXPIRES   LEVEL  PR BR RS AC
      CHARGE NO. PROJECT NUMBER

   1 OCMGL    IND. PUBLIC          27   84/05/19. 84/05/19. 84/05/19.
                  0         WRITE       12.05.23. 12.05.23. 12.05.23.
                       N  Y  D  Y
      2222       666N666

   2 DIRECS   IND. PRIVATE          2   84/05/19. 84/05/19. 84/05/19.
                  0         READ        12.05.24. 12.05.24. 12.05.24.
                       N  Y  D  Y
      2222       666N666

   3 CMDERR   IND. PUBLIC           8   84/05/19. 84/05/19. 84/05/19.
                  0         READ        12.05.31. 12.05.31. 12.05.31.
                       N  Y  D  N
      2222       666N666

   4 INTRCM5  DIR. PUBLIC         155 * 84/05/19. 84/05/19. 84/05/19.
                  0         READ        12.05.55. 12.05.55. 12.05.55.
                       N  Y  D  N
      2222       666N666

        3 INDIRECT ACCESS FILES,     TOTAL PRUS =        37.
        1 DIRECT ACCESS FILE,        TOTAL PRUS =       155.
```

Figure 5-5. Full CATLIST Example

LISTING ACCESSES BY OTHER USERS

If you have allowed other users access to a private file (the PERMIT command is discussed earlier in this section), or if you have a semiprivate file, you can obtain a list of accesses made by other users. Included in this list are the names of the other users, the number of times each user has accessed the file, the permitted access mode for each user, and a date and time. The date and time indicate when the user last accessed the file. If the user has never acessed the file, the date and time indicate when the user was granted permission to use the file. The format of the CATLIST command that does this is:

```
CATLIST,LO=FP,FN=filename.
```

The example in figure 5-6 is a list of two user names that accessed private file MAY belonging to user APP42.

```
CATALOG OF APP42                    FM/NOSHOP 84/05/19  12.48.08.  PAGE  1
         FILE NAME CMDERR
USER NAME     PERM.  EXPIRES  ACCESSES    DATE    TIME

  1. KY63472  WRITE*                1   yy/mm/dd. 14.03.42
  2. ABC4524   READ*                1   yy/mm/dd. 12.46.18.

NOTE:  An * indicates that you have given the user permission to use the file with
       the PERMIT command.
```

Figure 5-6. Listing Accesses by Other Users to a Private File

MAGNETIC TAPE PROCESSING 6

Magnetic tapes are a means of storing information. As with disk storage, information is stored in the form of files. The files are written on reels of tape. Each reel is called a volume; a volume generally contains a single file. Tapes offer the following advantages when compared to disk storage.

- Tapes are easily transported. Once you have written a tape, you can carry the tape to another computer, or to another installation, and read it.
- Tapes provide a convenient means of backing up or preserving a permanent file.
- Tapes are less expensive than the equivalent amount of disk storage.

A disadvantage of tapes is that processing is significantly slower than disk processing, because tapes require operator involvement. In deciding whether you should use tapes, consider the particular application you are using.

You must have specific validation to use magnetic tapes. You can use the LIMITS command to determine whether or not you are validated to use tapes and how many tape units you may have assigned to your job at one time.

When a tape comes from the factory, it is just a reel of 1/2-inch magnetic tape. Usually the site operator makes each tape unique with a process called blank labeling. Blank labeling puts identifying information on the tape so that it cannot be mistaken for another tape. In this section we assume that you are using a tape that has a label.

When blank labeling a tape, the system writes a volume header label (VOL1) at the beginning of the tape. This label has the family and user names of the tape owner and a unique name, called the volume serial number (VSN), to be associated with the tape. Blank labeling also determines both the type of tape unit and the tape density that is to be used to read from and write on the tape.

For easy identification, it is a good idea to attach a label to the tape reel that shows whether the tape is 7-track or 9-track, the tape density, and the VSN. For example, if a tape must be used on a 9-track tape unit, has an 800-character-per-inch (cpi) density, and has the VSN AAC123, place:

```
9-track, 800-cpi, VSN=AAC123
```

on the tape's external label.

If you cannot tell which type of tape unit to use with the tape, the tape's density, or its volume serial number, do not use the tape. Return the tape to the operator. You can ask the operator either to mount the tape and get the information you require or to blank label the tape again. If the operator blank labels the tape, you lose any information that was on the tape.

READING A LABELED TAPE

To access the file on a tape, you must make it a local file. Using the following format of the LABEL command makes the tape file a local file and assigns it a file name, for the purpose of reading the file.

```
LABEL,lfn,VSN=vsn,D=den,F=format,LB=lb,PO=R,R.
```

Parameter	Description
lfn	Specifies the local file name you want associated with the tape file. The tape file becomes a nonprimary file. You use this file name on other commands when referring to the tape file. lfn must be the first parameter listed; all others can follow in any order.
VSN=vsn	Specifies the volume serial number that uniquely identifies the reel of tape (the VSN should be on the external label of the tape). If the tape is mounted before you enter the LABEL command, the system automatically assigns the tape with this VSN to your job. If the tape is not mounted, the operator compares the VSN on the command with the VSNs on the external labels of tapes at the console and mounts the tape you requested.
D=den	Specifies the density of data and the type of tape.

Tape units write lengthwise in either seven or nine parallel paths. A tape unit is either 7-track or 9-track; it cannot be switched from one to the other.

Density of data is the number of characters written per inch of length. It is measured in characters per inch (cpi).

den can be:

den	Description
LO	200-cpi, 7-track.
HI	556-cpi, 7-track.
HY	800-cpi, 7-track.
HD	800-cpi, 9-track.
PE	1600-cpi, 9-track.
GE	6250-cpi, 9-track.

You must read a tape at the same density and on the same type of tape unit as was used when the tape was written. This information should be on the label attached to the outside of the tape.

Parameter	Description
F=format	Specifies the format of the tape. format can be:

format	Description
I	Internal (default).
SI	NOS/BE internal.

You should read a tape in the format that it was written in. Most tapes written on NOS have an internal (I) format.

If a tape was written on a computer system not manufactured by Control Data, or if it was written on a different Control Data system (for example, one of the 3000 systems), special processing is required. This processing is not described in this guide; refer to Volume 3, System Commands for information on processing these tapes.

LB=lb Specifies the type of label on the tape (assigned when the tape was written.)

A tape need not be labeled, but we strongly recommend using labeled tapes. They provide greater security against accidental overwriting, and they are simpler to process, because they can be assigned to your job without operator intervention. Labels are either ANSI-standard or nonstandard. ANSI-standard labels are the system default and we recommend them for new tapes. lb can be:

lb	Description
KL	ANSI-standard label (default). The system aborts the job if you specify LB=KL when the label is not ANSI-standard.
KU	Unlabeled.
NS	Nonstandard label.

PO=R Specifies that you want to read the tape; the system, therefore, does not allow you to write on it. This avoids accidentally writing on your tape.

This parameter instructs the operator not to mount the tape with a write ring in (the write ring is a flexible ring that can be inserted in the tape reel). If the operator mounts the tape with a write ring in, the job is suspended until the operator remounts the tape correctly.

R Specifies that the system is only to read the tape's file header label (HDR1). Included on this label are the name of the system on which the file was created, the creation date of the file, and who can access the file. You specify this parameter when you do not want to change the label. A parameter that allows the system to rewrite the label using your instructions is described later. R is the default and therefore can be omitted.

Example:

Assume you have a tape that contains data to be used by a program. The program identifies the data by the name DATA1. The external tape label has the following information on it.

9-track, 1600-cpi, VSN=DA432

The tape was originally written on NOS, has an ANSI-standard label, and is in internal format. The following LABEL command assigns the tape to your job.

LABEL,DATA1,VSN=DA432,D=PE,PO=R.

When the tape is assigned to your job, the system responds:

NTxxx, ASSIGNED TO DATA1, VSN=DA432.

xxx is the equipment number of the tape unit where the tape is located.

In an interactive job, your job will wait until the tape is mounted and assigned to your job. Then this response is returned to the terminal. Depending on your site's hardware and workload, mounting the tape you need may take a significant amount of time. Also, some sites may restrict interactive tape use. Creating a submit file (discussed in section 2) is a good way to process tapes from a terminal.

Once the tape is assigned to your job, you can treat it as you would any sequential file. You can access records in the order in which they appear on the tape, but to replace, insert, or delete records from the tape file, you must rewrite the entire file. For easier access to records and the ability to rewrite records, you may want to copy a tape file to disk when using it in a job.

To copy files to and from tape, you can use the following COPY command format.

COPY,I=lfn_1,O=lfn_2,V=x,TC=tc,N=copycnt.

Parameter	Description
I=lfn_1	Specifies the name of the local file to copy from. NOS starts copying from the current position of the file, unless lfn_1 is the primary file or you specify the V=x parameter. Primary files are rewound before each operation.
O=lfn_2	Specifies the name of the local file to copy to. NOS starts the copy at the current position of the file, unless you specify the V=x parameter. If no local file exists with the name lfn_2, NOS creates a local file with that name.
V=x	Specifies optional one through seven alphanumeric characters. If specified, both files are rewound, copied, rewound, verified, and rewound. x cannot be zero.

Parameter	Description
TC=tc	Specifies the termination condition. This, in conjunction with N=copycnt, specifies when the copy operation ends. You can specify:

tc	Description
EOF	Stops copying after copying the number of EOFs specified in N=copycnt.
EOD	Stops copying after encountering the number of double EOFs specified in N=copycnt. A double EOF is an end-of-file followed immediately by another end-of-file. EOD is the default.
EOI	Copies to end-of-information.

Copying continues either until the termination condition is met or until EOI is encountered. If copying is terminated by a double EOF, the second EOF is detected on lfn_1, but is not written on lfn_2.

N=copycnt	Specifies the count used with the TC=tc parameter. The default is 1.

In figure 6-1, a user makes a file that is on a standard NOS tape a local file, copies the local file to disk, and then saves the local file for later use. The tape has the following characteristics written on its external label.

1600-cpi, 9-track, VSN=RU5334

```
COPFIL.
USER,username,password.
LABEL,TAPFILE,VSN=RU5334,D=PE,PO=R.
COPY,I=TAPFILE,O=DISFILE,V=V,TC=EOF.
SAVE,DISFILE.
```

Figure 6-1. Copying a Tape File to Disk

WRITING ON A LABELED TAPE

Before you can write information on a tape file, you must assign a file name to it, so that it is accessible to your job. You use the following format of the LABEL command.

```
LABEL,lfn,VSN=vsn,D=den,F=format,LB=lb,PO=W,R.
```

Parameter	Description
lfn	Specifies the local file name you want associated with the tape file. The tape file becomes a nonprimary file. You use this file name on other commands when referring to the information on the tape. lfn must be the first parameter; all others can follow in any order.
VSN=vsn	Specifies the volume serial number that uniquely identifies the reel of tape. Refer to Reading a Labeled Tape for more information.
D=den	Specifies the density of data and the type of tape unit. den can be:

den	Description
LO	200-cpi, 7-track.
HI	556-cpi, 7-track.
HY	800-cpi, 7-track.
HD	800-cpi, 9-track.
PE	1600-cpi, 9-track.
GE	6250-cpi, 9-track.

The density and the type of tape unit you must use are determined when the tape is first labeled. This information should be attached to the outside of the reel of tape. Refer to Reading a Labeled Tape for more information.

F=format	Specifies the format of the tape. When writing tapes, you use the default F=I.
LB=lb	Specifies the type of labels on the tape. When writing tapes, use the default LB=KL. This means that the tape has ANSI-standard labels.
PO=W	Specifies that you want to write on the tape. When you leave the reel with the operator, you should indicate that you want to write on the tape. If the operator mounts the tape without the write ring, the job is suspended until the operator remounts the tape correctly.
R	Allows no changes to the HDR1 label except the updating of the tape's creation date when PO=W is also specified. In most cases, you do not need to be interested in the information on this label and, therefore, do not need to change it when a new file is written on the tape. R is the default and can therefore be omitted. The parameter that allows the system to rewrite this label is described with multifile sets later in this section.

After the tape file has been assigned to your job, you can write on it as you would write on any local file. The example in figure 6-2 shows how to copy a file from mass storage to tape.

Batch Job	Description
JOBTAPE. USER,username,password.	Job identification.
ATTACH,DRILBIT.	The user makes the direct access file named DRILBIT a local file.
LABEL,DRBIT,VSN=DR1450,D=GE,PO=W.	The user assigns the file name DRBIT to the tape with the VSN DR1450 and specifies the following: the density is 6250 cpi; the tape is nine-track (D=GE); information written on the tape file is to be in internal format (F=I is the default); the tape label is ANSI-standard (LB=KL is the default); and the tape must be mounted with the write ring in (PO=W).
COPY,I=DRILBIT,O=DRBIT,V=VER,TC=EOI.	The user copies the complete direct access file DRILBIT to the tape file DRBIT, rewinds, and verifies that the files are the same.
--EOI--	End-of-information.

Figure 6-2. Copying a File to Tape

The example in figure 6-3 shows how to add an input record to the end of a tape file that contains three records.

Batch Job	Description
MUSJOB. USER,username,password.	Job identification.
LABEL,MUSIC,VSN=MUS77,D=HY,PO=W.	The user assigns file name MUSIC to the tape with the VSN MUS77 and specifies the following: the density is 800 cpi; the tape is seven-track (D=HY); the format is internal (F=I is the default); the label is ANSI-standard (LB=KL is the default); and the tape must be mounted with the write ring in (PO=W).
SKIPR,MUSIC,3.	The user skips the first three records on the tape.
COPYBR,,MUSIC.	The user copies one record of data from the next record in the batch job to the current position of the tape. All information that is on the tape after the current position is lost.
REWIND,MUSIC.	The user rewinds the tape.
COPYSBF,MUSIC.	The user prints the complete file to see what is actually on the tape.
--EOR--	End-of-record.
NOTE 1 NOTE 2 NOTE 3 NOTE 100 NOTE 101 NOTE 102	The user copies this data to the tape file as the fourth record.
--EOI--	End-of-information.

Figure 6-3. Adding a Record to a Tape File

COPYING FROM ONE TAPE TO ANOTHER TAPE

Normally, the system allows you to use only one tape unit at a time during a job. If your job requires more than one tape unit to be used concurrently, you must be authorized to do so by site personnel, and you must include the RESOURC command before your request for the tape units. To find out how many tape units you are authorized to use concurrently, you enter the LIMITS command (discussed in Using System Resources in section 2). The RESOURC command's format is:

RESOURC,rt_1=u_1,rt_2=u_2,...,rt_n=u_n.

Option	Description
rt_i	The type of tape unit the job needs. rt_i can be one or more of:

rt	Description
MT	200-, 556-, or 800-cpi, 7-track.
HD	800-cpi, 9-track.
PE	1600-cpi, 9-track.
GE	6250-cpi, 9-track.

u_i	The maximum number of tape units needed concurrently that are of the type named by rt.

For example, if a job needs to use one 800-cpi, 7-track tape and two 1600-cpi, 9-track tapes concurrently, you use the following RESOURC command.

```
RESOURC,MT=1,PE=2.
```

If you specify more tape units than you are authorized to use concurrently, you get the following message.

```
DEMAND VALIDATION ERROR.
```

You must either change the job so that it uses fewer tape units concurrently, or talk to site personnel about increasing the number of tape units you are authorized to use.

The example in figure 6-4 shows how to copy a file from one tape to another tape.

Batch Job	Description
TRANTAP. USER,username,password.	Job identification.
RESOURC,HD=2.	The user specifies that the job will require two nine-track tape units that read and write 800 cpi.
LABEL,OLDTAP,VSN=ABCD,D=HD,PO=R.	The user assigns file name OLDTAP to the tape that contains the file to be copied. The tape must conform to the type of tape units assigned with the RESOURC command, in this case, 800-cpi and nine-track.
LABEL,NEWTAP,VSN=1234,D=HD,PO=W.	The user assigns file name NEWTAP to the tape that the file will be copied to and specifies PO=W in order to write on the tape. This tape must be an 800-cpi, nine-track tape.
COPYEI,OLDTAP,NEWTAP,V.	The user copies the file from OLDTAP to NEWTAP and verifies that the files are the same.
--EOI--	End-of-information.

Figure 6-4. Copying from One Tape to Another Tape

After you are finished using a tape file, you can release it from your job. Releasing a tape file means that you cannot access the file again unless you enter another LABEL command. The RETURN and UNLOAD commands both release tape files, but the RETURN command may also decrease the number of tapes you scheduled for your job when you entered the RESOURC command. We recommend that you use the UNLOAD command to release tape files if you are going to use more tape files in the job.

The format of the UNLOAD command is shown in section 4 under Releasing Local Files. When the system processes the UNLOAD command, it informs the operator that the tape can be removed from the tape unit to make room for another tape to be used later in the job. Figure 6-5 shows the use of the UNLOAD command in creating a new tape file from two other tape files.

Batch Job	Description
NEWFILE. USER,username,password.	Job identification.
RESOURC,PE=2.	The user allows the job to use two 9-track, 1600-cpi tape units concurrently.
LABEL,MASTER,VSN=MA4324,D=PE,PO=W.	The user assigns the tape with the VSN MA4324 as file MASTER and specifies that it must be mounted with the write ring in (PO=W). This is the tape that will contain the new file.
LABEL,SECTION,VSN=SECT,D=PE,PO=R.	The user assigns the tape with the VSN SECT as file SECTION and specifies that the tape must be mounted without the write ring (PO=R). This tape contains a record to be copied to file MASTER.
COPYBR,SECTION,MASTER.	The user copies the first record on file SECTION to file MASTER.
UNLOAD,SECTION.	The user releases local file SECTION. This releases the tape that is associated with file SECTION but does not change the job's tape resources. Only one tape is now assigned to the job.
LABEL,BLOCK,VSN=BL131,D=PE,PO=R.	The user assigns the tape with the VSN BL131 as file BLOCK and specifies that the tape must be mounted without the write ring. This tape contains two records to be copied to file MASTER.
COPYBR,BLOCK,MASTER,2.	The user copies the first two records on file BLOCK to file MASTER. The tape with VSN MA4324 now has three records on it.
--EOI--	End-of-information.

Figure 6-5. Tape Copying with Limited Resources

MULTIVOLUME FILES

To access a file that spans more than one reel of tape, you should specify the VSNs of all the tapes that the file is on. You use the VSN=vsn parameter on the LABEL command as follows:

VSN=vsn_1/vsn_2/.../vsn_n

Option	Description
vsn_i	The VSNs of the tapes that contain the file you want to access. The system accesses the tapes in the order that their VSNs appear.

For example, a file is contained on two tapes with the VSNs T15 and T16. The tape with the VSN T15 contains the first part of the file, and the tape with the VSN T16 contains the end of the file. The tapes also have the following characteristics: ANSI-standard labels, 800 cpi, 9-track, and internal format. The following LABEL command accesses the file by the name BOOK.

```
LABEL,BOOK,VSN=T15/T16,D=HD,PO=R.
```

If, after reading the file, you want to position the file at its beginning, you can enter the REWIND command. The system then notifies the operator to mount the first reel of the tape.

When writing a file to tape, you should consider the possibility that the file may not fit on one tape. The amount of information that can be written on a tape cannot be determined with certainty; it is dependent on the condition of the tape, the tape unit, and the density at which the file is written. If you are not sure that one tape will hold the file, leave more than one tape with the operator and specify the VSNs for all the tapes on the VSN=vsn parameter of the LABEL command. You should put the VSNs in the order you want the tapes used. The system will ask the operator to mount the tape with the second VSN only after the first tape is full.

If you specify only one VSN and your file does not fit on one tape, the operator must supply another tape to finish writing your file. This may cause problems for you, since this other tape is not yours; it may cause problems for the operators, since they must supply a tape in order to finish your job. The best way to avoid these problems is to anticipate when you might need more than one tape and to specify multiple VSNs on the LABEL command.

NOTE

Most commands have a maximum length of 80 characters. An exception is a LABEL command used noninteractively. In this case, the LABEL command can span more than one line, each line having 80 or fewer characters, and the last line ending with a termination character. If you want to use the LABEL command interactively and it is more than 80 characters, either you can shorten it by separately specifying the VSN(s) with the VSN command (described in Volume 3, System Commands), or you can put the command in a procedure (procedures are discussed in section 8).

An example of writing a large file to tape is shown in figure 6-6.

Batch Job	Description
`LGFILE.` `USER,username,password.`	Job identification.
`ATTACH,LARGE.`	
`LABEL,BIG,VSN=MA14/MA15,D=PE,PO=W.`	The user assigns the tape with VSN MA14 to the job as local file BIG and also specifies that: • The tape with the VSN MA15 is to be used if the file does not fit on the first tape (VSN=MA14/MA15). • The system can write on the tape (PO=W) in 1600 cpi (D=PE) and internal format (the default format). • The tape's labels are ANSI-standard.
`COPYEI,LARGE,BIG,V.`	The user copies the complete file LARGE to tape and verifies that the files are the same. If the system reaches the end of the first tape before reaching EOI in file LARGE, the system asks the operator to mount the next tape (VSN MA15). After that tape is mounted the system continues copying the file until it reaches EOI.
`--EOI--`	End-of-information.

Figure 6-6. Writing a Multivolume File

MULTIFILE SETS

You can put more than one file on a tape and still access each file separately. This type of file arrangement on tape is called a multifile set. Each file has its own HDR1 label which uniquely identifies it and also associates it with the other files on the tape.

A multifile set differs from a multifile file.

- **A multifile file is more than one file under one file name. When you access the multifile file, you can access all files within it (multifile files are described in section 3).**
- A multifile set is more than one file, each with a unique name, on a tape. When you access the tape, you can access only one file on it at a time.

You must decide, before you put a file on tape, if you want other files to reside on the same tape. If you do, you must assign a set identifier to this first file when you write the file to tape. You specify the file set identifier on the LABEL command with the following parameter.

SI=setid

Option	Description
setid	**A 1- to 6-character alphanumeric set identifier chosen by you. We recommend that you pick a setid that is easy to remember, since you have to specify it when adding other files to the tape.**

Each tape can have only one set identifier.

When a tape is blank labeled, the information written on the HDR1 label does not allow the tape to be used as a multifile set tape (that is, you cannot specify a set identifier). Therefore, you must have the system rewrite the HDR1 label by specifying W on the LABEL command, rather than using the default value R (R allows the system to read the HDR1 label, and if you also specify PO=W, to change the creation date of a file). W tells the system to rewrite the HDR1 label using the information you provide in other LABEL parameters.

Before writing the first file of a multifile set on a tape that has been blank labeled, you would enter a LABEL command similar to:

```
LABEL,A,VSN=PAR42,SI=PAR,D=PE,PO=W,W.
```

Responding to this command, the system rewrites the HDR1 label for the first file that is to be written to the tape with VSN PAR42. The set identifier PAR allows you to write more than one file to the tape.

In a multifile set, the system assigns number 0001 to the first file, 0002 to the second file, and so on. To add another file to a multifile set, you must specify:

```
QN=9999
```

as well as the SI=setid parameter on the LABEL command. This parameter positions the tape at the end of the last file and allows you to write a new file to tape. The system then assigns the next available sequence number to the new file. For example, if the file added is the fifth file on the tape, it is assigned sequence number 5. If you do not specify QN=9999, the system assumes QN=1 and positions the new file as the first file of the set.

You might enter the following commands to add file G to a tape with VSN COPY18 that already has a file on it.

```
GET,G.
LABEL,B,VSN=COPY18,SI=COPY,D=PE,PO=W,W,QN=9999.
COPYEI,G,B.
```

The GET command makes file G a local file. The LABEL command assigns the tape to the job and positions it so that you can add a new file to the tape. The COPYEI command then causes file G to be written on the tape.

If you want to read a file that is in a multifile set and you remember the position of the file, you can access it by specifying:

```
QN=seqno
```

on the LABEL command. seqno is the sequence number assigned by the system. For example, if you know the file you want to read is the third file in a multifile set with the set identifier RTV on a tape with the VSN RTV435 and a density of 6250, you can use the following LABEL command to assign the file to your job.

```
LABEL,B,VSN=RTV435,SI=RTV,D=GE,PO=R,QN=3.
```

When writing a file to tape, you can add a descriptive identifier to the HDR1 label, which may help you access the file without remembering its position. You add the identifier by specifying the following parameter on the LABEL command.

```
FI=fileid
```

Option	Description
fileid	A 1- to 17-character file identifier recorded in the HDR1 label. If fileid contains nonalphanumeric characters, it must be enclosed in literal delimiters ($). The default is blank.

For example, with the following LABEL command, you can access a tape with the VSN BOX13, the setid BOX, and a density of 1600 with the intention of writing a new file identified as MODEL 342 on the tape.

```
LABEL,C,VSN=BOX13,SI=BOX,FI=MODEL342,D=PE,PO=W,W,QN=9999.
```

To subsequently read the file, you would use the following LABEL command.

```
LABEL,D,VSN=BOX13,SI=BOX,FI=MODEL342,D=PE,PO=R.
```

The system searches the tape for a file with an identifier of MODEL342. If you want to position the tape at the beginning of the file identified by the LABEL command, you can enter the REWIND command. This rewinds the tape to the beginning of that file.

You can access only one file at a time on any given tape. To access different file on the same tape, you must enter another LABEL command. This then repositions the tape to the new file requested.

The following set of commands adds a new file to a multifile set and then prints the file.

```
GET,LIST.
LABEL,E,VSN=STIR4,SI=STIR,FI=TERMS,D=PE,PO=W,W,QN=9999.
COPYEI,LIST,E.
REWIND,E.
COPYEI,E.
```

For more examples using multifile sets, refer to the discussion of the LABEL command in Volume 3, System Commands.

Despite all of your good intentions, you may find that you have a tape with files on it and you cannot recall how many files are on the tape nor how you have identified the files. With the help of the LISTLB command, you can refresh your memory. This command lists the contents of ANSI-standard labels. Included in the list are the VOL1 label, which is at the beginning of the tape, and the HDR1 label, which begins each file on the tape. Each HDR1 label contains the file identifier, the set identifier, and the sequence number that shows the position of each file on the tape. The LISTLB command's format is:

LISTLB,lfn,SI=setid.

Parameter	Description
lfn	The name assigned to the tape in the LABEL command.
SI=setid	The set identifier you entered on the LABEL command when you put the first file on the tape.

If you did not specify a set identifier on the LABEL command, you can enter:

LISTLB,lfn.

This will give you the labels of one file on the tape.

To use the LISTLB command the tape must be assigned to your job and positioned at its beginning. The example in figure 6-7 shows the commands used in listing the labels on a tape that contains two files. The example also shows the output from the LISTLB command, highlighting the information relevant to this discussion. Other information in the output pertains to aspects of tape management that are not discussed in this guide (refer to Volume 3, System Commands).

If you did not know the set identifier, you could enter the following LISTLB command in place of the one shown in figure 6-7.

LISTLB,T,LO=H.

When you use the LO=H parameter, the system returns only one HDR1 label. From the information on this label, you then would know the set identifier and could enter

LISTLB,T,SI=DUMP1.

to see the labels in the second file.

Once you have used the LISTLB command with the SI=setid parameter, you cannot perform any operations on the tape other than to release the tape unless you enter another LABEL command.

```
LABEL,T,VSN=DUMP,D=PE,PO=R.
LISTLB,T,SI=DUMP1.
```

```
     LISTLB - LIST MAGNETIC TAPE LABELS.                                  PAGE  1

    VOL1 LABEL READ:  VOL1DUMP                                             NOS

     VSN=DUMP   , VA=  , OWNER ID=NOSCL SHBJK2201, LSL=1.

    HDR1 LABEL READ:     HDR1UPGRADE                    DUMP1 0001000100001

     FI=UPGRADE            , SI=DUMP1   ,  SN=0001, QN=0001, G=0001,

    EOF1 LABEL READ:     EOF1UPGRADE                    DUMP1 0001000100001

     FI=UPGRADE            , SI=DUMP1   ,  SN=0001, QN=0001, G=0001,

    HDR1 LABEL READ:     HDR1CORRECT1                   DUMP1 0001000200001

     FI=CORRECT1           , SI=DUMP1   ,  SN=0001, QN=0002, G=0001,

    EOF1 LABEL READ:     EOF1CORRECT1                   DUMP1 0001000200001

     FI=CORRECT1           , SI=DUMP1   ,  SN=0001, QN=0002, G=0001,

            5 LABELS READ.                                5 LABELS PRINTED.
```

Field	Description
FI	File identifier.
SI	Set identifier.
QN	Sequence number.

Figure 6-7. Listing Labels

ERROR CONTROL 7

When you are using the system interactively, an error message is returned to the terminal when you enter an incorrect command. You can then correct any mistake you may have made simply by reentering the command correctly. This is not possible in a batch jobs or in NOS procedures (described in Section 9); the system automatically aborts the job or procedure when an error occurs. However, you can create a section containing commands that ensure either that the results from the completed part of the job are saved or that the files are kept as they were before the job. You put these error control commands after all the other commands in the first record of the job. They must be preceded by the EXIT command. Its format is:

```
EXIT.
```

When an error occurs, the system searches the remaining commands in the first record of the job for the EXIT command and then processes the commands after it. If no error occurs, the job ends when it reaches the EXIT command.

Figure 7-1 is an example of a batch job that contains error control commands.

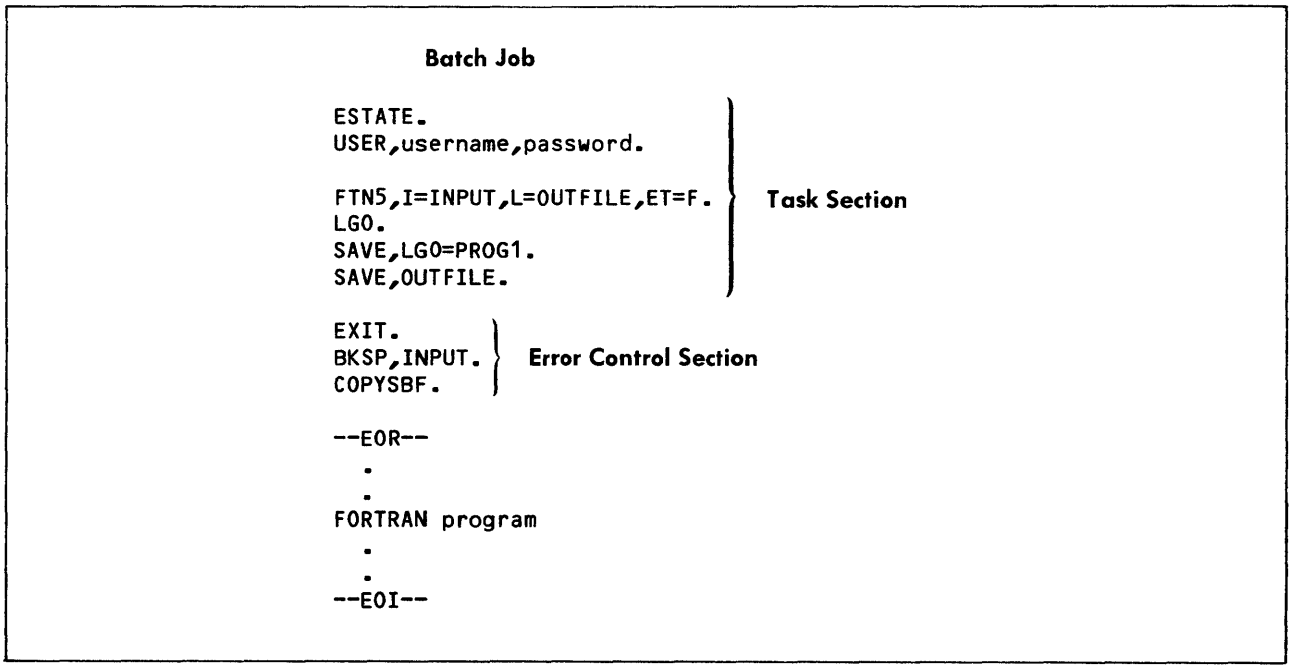


Figure 7-1. Batch Job Error Control Commands

If an error occurs in compiling the FORTRAN program shown in the job in figure 7-1, the commands boxed in figure 7-2 under Compile Error are processed. If no errors occur, the commands boxed in figure 7-2 under No Errors are processed.

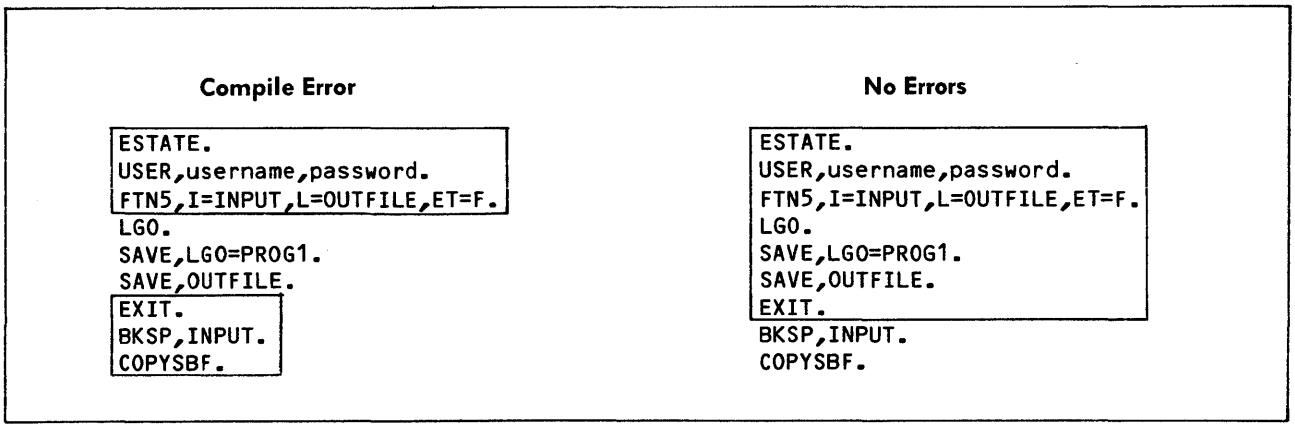


Figure 7-2. Exit Processing

If you want the system to attempt to process each command in a group of commands regardless of any errors, you can use the NOEXIT command. Its format is:

```
NOEXIT.
```

If you put this command before the commands you want processed, the system does not search for an EXIT command when an error occurs. Instead, it processes each command following NOEXIT, even if an error occurs in a previous command (refer to figure 7-3).

Batch Job	Description
NOSTATE. USER,username,password.	Job identification.
NOEXIT.	The user indicates that the following commands are to be processed even if errors occur.
GET,INFILE.	The user makes INFILE a local file.
FTN5,I=INFILE.	The user calls FORTRAN to compile INFILE and then execute LGO.
REPLACE,LGO=KEY. --EOI--	The user replaces the information in permanent file KEY with the information in local file LGO.

Figure 7-3. Example of NOEXIT Command

You may want some commands to be processed regardless of previous errors, and other commands to be processed only if no errors occur. The ONEXIT command reverses the effect of a NOEXIT. That is, if an error occurs the system searches the commands after the command that produced the error for an EXIT command. If an EXIT command is not found, the job aborts. The ONEXIT command's format is:

```
ONEXIT.
```

Example:

The following batch job appends file DECK to file CARD, rewinds file CARD, and then prints file CARD.

```
JOBEXIT.
USER,username,password.
NOEXIT.
GET,CARD.
GET,DECK.
APPEND,CARD,DECK.
ONEXIT.
REWIND,CARD.
COPYSBF,CARD.
EXIT.
CATLIST.
SAVE,CARD=PLAYER.
--EOI--
```

If an error occurs when the system is processing:

```
GET,CARD.
GET,DECK.
APPEND,CARD,DECK.
```

it continues with the next command. For example, if file DECK does not exist, processing the command GET,DECK produces an error, but the system continues with the APPEND command.

Upon reaching the ONEXIT command, the system starts reacting to errors. If an error occurs when processing:

```
REWIND,CARD.
COPYSBF,CARD.
```

the system searches for the EXIT command and processes those commands after it.

JOB'S DAYFILE

The system keeps a complete history of your job's interaction with the system. You can check this history, called the job dayfile, to see what happened during your job. It includes both the commands processed in the job and any messages sent to your job. Each line in the body of the dayfile has the following format.

```
hh.mm.ss.message
```

hh.mm.ss. is the time of day the command or message was placed in the dayfile.

Many messages are the result of errors that occur when the system attempts to process commands. Explanations for these messages are in either appendix B of Volume 3, System Commands or the manual that describes the product you were attempting to use when the message appeared.

In a batch job, the dayfile is printed at the end of your job. A dayfile printed with a batch job includes a header, which gives the job name and date. The last few lines of a batch dayfile show the type and amount of system resources the job used. Figure 7-4 shows a batch job and the dayfile associated with it.

Batch Job

```
FILEDAY.
USER,BMK2804,KKKK.
CHARGE,*.
GET,BATEX.
APPEND,DAY,BATEX.
GET,DAY.
COPYEI,DAY,NEWDAY.
SAVE,NEWDAY.
```

Dayfile

```
ABNS    CDC NETWORK OPERATING SYSTEM.       NOS 2                  08.11.32. yy/mm/dd.
08.11.31.FILEDAY.
08.11.31.UCCR, 06, 024,
08.11.31.USER,BMK2804,.
08.11.31.GET,BATEX.
08.11.31.APPEND,DAY,BATEX.
08.11.32.GET,DAY.
08.11.32.COPYEI,DAY,NEWDAY.
08.11.32. EOI ENCOUNTERED.
08.11.32.SAVE,NEWDAY.
08.11.32.UEAD,      0.002KUNS.
08.11.32.UEPF,       0.043KUNS.
08.11.32.UEMS,      0.215KUNS.
08.11.32.UECP,      0.009SECS.
08.11.32.AESR,      2.020UNTS.
08.11.32.$OUT(*/OP=E)
08.11.32. NO FILES PROCESSED.
08.11.32.$DAYFILE(OUTPUT,JT=D)
08.11.56.UCLP, 42, 070,     0.128KLNS.
```

Figure 7-4. Dayfile from a Batch Job

In an interactive job, some messages that go into the dayfile are also printed at the terminal as the error occurs. The terminal message may be all the information you need to recognize what error occurred. However, you can obtain the entire dayfile by entering the DAYFILE command. Its format is:

DAYFILE,lfn.

Parameter	Description
lfn	The name of the local file on which the dayfile is to be written. If lfn is omitted, file OUTPUT is assumed and the dayfile is displayed at the terminal.

The system returns a complete history of your interactive job from the time you logged in until you entered the DAYFILE command on the file named (refer to figure 7-5).

```
batch
RFL,0.
/get,dsy
 DSY NOT FOUND.
/get,day
/copycf,day,apple,2
 EOI ENCOUNTERED.
/rewind,*
    5 FILES PROCESSED.
/save,apple
/dayfile
 08.39.45.AJAI.
 08.39.45.USER,BMF2804,,NOSHOP.
 08.39.46.RECOVER,OP=T.
 08.42.09.RFL,0.
 08.42.21.GET,DSY.
 08.42.21. DSY NOT FOUND.
 08.42.29.GET,DAY.
 08.42.57.COPYCF,DAY,APPLE,2.
 08.42.57. EOI ENCOUNTERED.
 08.43.12.REWIND,*.
 08.43.12.   5 FILES PROCESSED.
 08.43.29.SAVE,APPLE.
 08.43.38.DAYFILE.
 USER DAYFILE PROCESSED.
```

Figure 7-5. Interactive Dayfile Example

After having the dayfile printed, you can continue entering other commands. Later in the session, entering DAYFILE causes the system to display those dayfile entries made since you entered the last DAYFILE command.

In a job submitted with a ROUTE or SUBMIT command, you can use the DAYFILE command to check the results of the job without having to retrieve the printout at the central site. For example, you can add the following commands to the end of the command record.

```
DAYFILE,DAY.
REPLACE,DAY.
EXIT.
DAYFILE,DAY.
REPLACE,DAY.
```

The system puts the dayfile on file DAY and then makes the file permanent. This happens whether or not errors are made in processing any of the other commands in the job (the EXIT command is described earlier in this section). After you submit the job you can list file DAY at the terminal to see if the job was processed correctly. In the example in figure 7-6, the submit job has a command with an error in it. Therefore, the system processes:

```
EXIT.
DAYFILE,DAY.
REPLACE,DAY.
```

After finding the error, you can correct it and submit the job again.

File ERROREX

```
/JOB
ERRORS.
GER,EXAMPLE.            ***ERROR***
FTN5,I=EXAMPLE,L=0,SEQ.
LGO.
DAYFILE,DAY1.
REPLACE,DAY1.
EXIT.
DAYFILE,DAY1.
REPLACE,DAY1.
```

Terminal Session	Description
/get,errorex	The user gets and submits file ERROREX.
/submit,errorex,bc	
08.46.22. SUBMIT COMPLETE. JOBNAME IS AAJS	
/get,day1	The user checks the dayfile.
/copy,day1	
08.46.22.ERRORS.	
08.46.22.USER,BMK2804,.	
08.46.22.GER,EXAMPLE. ***ERROR***	The job did not complete because of an error in GER,EXAMPLE.
08.46.22. INCORRECT COMMAND.	
08.46.22.EXIT.	
08.46.22.DAYFILE,DAY1.	
EOI ENCOUNTERED.	

Figure 7-6. Using the DAYFILE Command in a Submitted Job

ERROR CONTROL BY INDIVIDUAL COMMANDS

The no-abort parameter can be added to the APPEND, ATTACH, DEFINE, GET, OLD, and PURGE commands to allow job processing to continue if the system encounters errors in processing these commands. Usually when an error occurs in a batch job, the system searches the job for an EXIT command. If one is not found, the job aborts. When the NA parameter is present on a command, the job continues even if an error occurs when the system processes the command. The no-abort parameter's format is:

```
NA
```

The command formats including the NA parameter are:

```
APPEND,pfn,lfn1,lfn2,...,lfnn/NA.

ATTACH,lfn=pfn/NA.

DEFINE,lfn=pfn/NA.

GET,lfn=pfn/NA.

OLD,lfn=pfn/NA.

PURGE,pfn1,pfn2,...,pfnn/NA.
```

Explanations of the other parameters on the commands are in sections 4 and 5. If you use the NA parameter and an error occurs, the error is handled in one of two ways.

- If the error condition is temporary, the job will wait until the condition is corrected. For example, if you request a direct access file and the file is being accessed by another job, your job is suspended until the file is available.
- If the error condition is not temporary, your job continues with the next operation. For example, if you request a file that does not exist, your job continues with the next command in your job file. However, any following command that used the file would cause an error, since the file does not exist.

If you use the NA parameter on a command entered interactively and are not in the batch subsystem, the system does not return a message if the command causes an error. You must enter either the DAYFILE or ENQUIRE,F command to determine whether the command was processed correctly. ENQUIRE,F lists the local files associated with the job at the time the command is executed.

You may want the system to suspend your job if the condition is a busy file, but to abort your job if the condition is not temporary. The wait-if-busy (WB) parameter gives you this choice. You can specify WB in place of NA on the APPEND, ATTACH, DEFINE, GET, OLD, and PURGE commands to establish the wait-if-busy option for a particular file.

ERRORS DETECTED BY THE HARDWARE

When the hardware detects an error condition during an interactive job, the system terminates the command and dumps the job's exchange package and a portion of the job's central memory to a local file called ZZZDUMP. This information may help you determine what error occurred. The system does not rewind ZZZDUMP before or after the dump. The following message is printed at the terminal.

```
EXCHANGE PACKAGE/MEMORY DUMP ON FILE ZZZDUMP.
```

To examine the information you must rewind and list file ZZZDUMP. Refer to appendix D for an explanation of the exchange package and error conditions.

FLOW CONTROL COMMANDS 8

As you have seen in the previous sections, an interactive NOS session usually consists of a user entering individual commands on a line by line basis. If you are a typical user, however, you will also from time to time find yourself writing sequences of commands to perform a specific task or tasks. Sequences of commands occur in batch jobs and in NOS procedures. (We define NOS procedures in this and the following section.) In either case, the sequence of commands is placed in a file such that you can execute the batch job or procedure file in much the same way that you would execute a program file.

Most programming languages include structuring tools to allow a program to control the execution sequence of program statements. Examples of structuring tools include testing for conditions, branching or skipping program statements, using counters and iterative loops, and calling subroutines. The purpose of NOS Flow Control commands is to provide similar structuring tools for NOS jobs.

The Flow Control commands are a subset of NOS commands that enable your job to:

- conditionally or unconditionally skip a sequence of NOS commands.
- cycle through a loop of NOS commands until a specified condition occurs.
- call subroutine-like sequences of NOS commands called procedures.

In this section we describe the Flow Control commands and command syntax, as well as the techniques used for skipping or looping through sequences of commands. We talk about how to write and use NOS procedures in Section 9.

The following discussions will give you a conceptual understanding of skips, loops, and procedures as they are implemented in NOS jobs. The functions and formats of all the Flow Control commands are summarized in Table 8-1.

SKIPPING COMMANDS

NOS has two skipping commands, the SKIP command and the IF command.

The SKIP command initiates an unconditional skip. In other words, the system initiates the skipping of commands every time the SKIP command is encountered.

The IF command allows your job to skip commands on a conditional basis. Depending on the status of system conditions that you define within the IF command itself, the IF command causes the system to either execute or skip the following command or sequence of commands.

SKIP AND ENDIF COMMANDS

The SKIP command works together with the ENDIF command to unconditionally skip a sequence of one or more NOS commands. The SKIP and ENDIF command formats are:

```
SKIP,label.
ENDIF,label.
```

The label field is a string of up to ten alphanumeric characters that must begin with an alphabetic character. The SKIP and ENDIF commands must have identical labels.

When a SKIP command is encountered, the system unconditionally skips all commands that occur between the SKIP command and the matching ENDIF command. For example, in the following sequence of commands, the SKIP command ensures that the RETURN command will not be executed:

```
SKIP,BT.
RETURN,*.
ENDIF,BT.
```

IF AND ENDIF COMMANDS

The IF and ENDIF commands work together to initiate and terminate conditional skipping of a sequence of commands. The IF command format is:

```
IF,expression,label.
```

where expression is a logical expression that can be evaluated to a true or false conclusion, and label is a string of up to ten alphanumeric characters beginning with an alphabetic character. IF and ENDIF must have identical labels.

IF and ENDIF work together in the same way as SKIP and ENDIF except that the IF command contains a logical or arithmetic expression that can be evaluated to a true or false conclusion. If the expression is false at the time the IF command is processed, the system skips all commands that appear between IF and ENDIF. If the expression is true, all commands are executed in their original order. Figure 8-1 shows the action of the IF command in diagram form.

As an example of the IF command, look at the following sequence of commands used to perform a copy operation.

```
IF,FILE(SOURCE,EOI),LABEL1.
REWIND,SOURCE.
ENDIF,LABEL1.
COPYSBF,SOURCE,OUTPUT.
```

The IF command in the above example contains the expression FILE(SOURCE,EOI). Later in this section we explain the symbols and formats used to construct expressions, but for now it is sufficient to understand that this expression states that file SOURCE is currently positioned at its end-of-information. In this sequence of commands, if the expression is true at the time the IF command is processed, then file SOURCE will be rewound before the copy operation. If the expression is false, copying will begin at the current position within file SOURCE.

Table 8-1. Flow Control Commands

Command	Description
SKIP,label.	Unconditionally initiates skipping of one or more NOS commands.
IF,expression,label.	Conditionally initiates skipping of one or more NOS commands.
ELSE,label.	Initiates skipping of commands if the expression in the matching IF command is true; terminates skipping if the expression is false.
ENDIF,label.	Terminates skipping initiated by a matching SKIP, IF, or ELSE command.
WHILE,expression,label.	Establishes the beginning of a loop. If the expression in the WHILE command is true, the loop is processed; if the expression is false, it is not processed.
ENDW,label.	Marks the end of a loop established by a matching WHILE command.
SET,expression.	Assigns values to NOS symbolic names.
DISPLAY,expression.	Evaluates an expression and displays the result in your job's dayfile.
REVERT,comment or REVERT,NOLIST. or REVERT,ABORT.comment	Terminates a procedure and returns control to the calling job or procedure. (REVERT is described in Section 9.)
BEGIN,pname,pfile,p_1,p_2,...,p_n or pname,p_1,p_2,...,p_n.	Calls a NOS procedure. (BEGIN is described in Section 9.)

IF, ELSE, AND ENDIF COMMANDS

Sometimes you might want to write two alternate sequences of commands such that one of the sequences will be executed if the expression is true and the other sequence will be executed if the expression is false. You can do this using the ELSE command with IF and ENDIF.

The ELSE command acts as a switch that either suspends or resumes the processing of commands. If the expression in the IF command is true, the system continues processing commands until it encounters the ELSE command. At that point, the system suspends processing until it comes to the ENDIF command. On the other hand, if the expression in the IF command is false, the system immediately suspends processing and jumps to the matching ELSE command. The system then resumes processing with the next command following ELSE. Figure 8-2 shows the action of IF, ELSE, and ENDIF in diagram form.

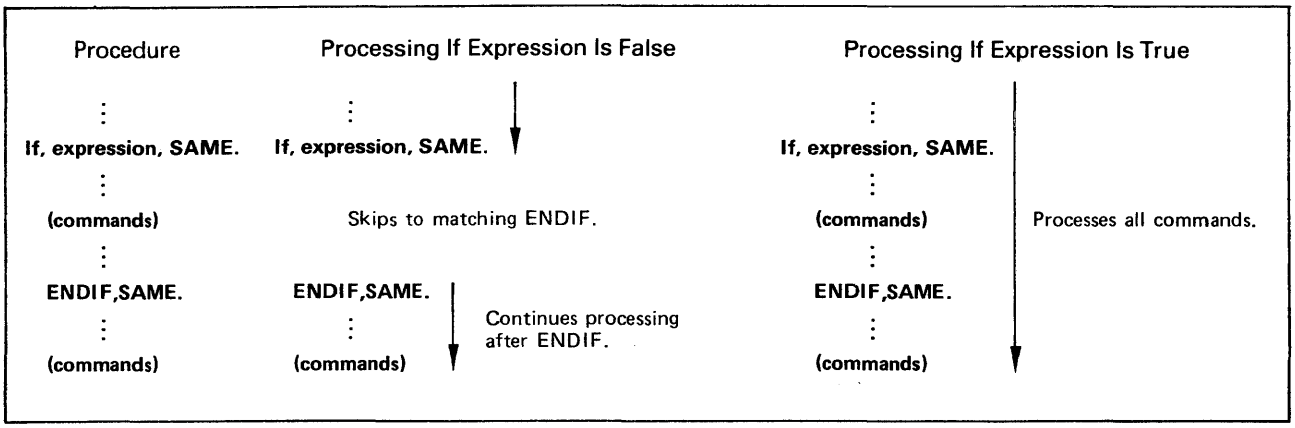


Figure 8-1. IF/ENDIF Processing

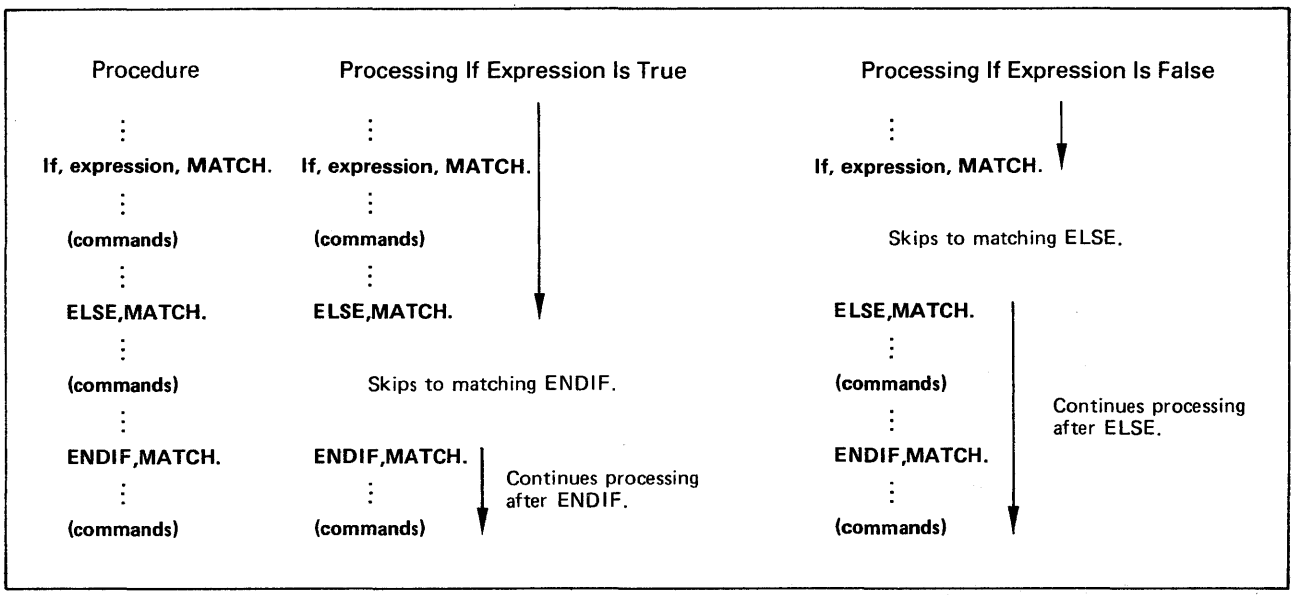


Figure 8-2. IF/ELSE/ENDIF Processing

As an example of the ELSE command, assume that you want to write an IF command to determine whether or not file WATCH is local to your job. Assume also that you want to rewind WATCH if it is local and that you want to get it from your permanent file if it is not local. The following sequence of commands accomplishes these objectives:

```
IF,FILE(WATCH,LO),L1.
REWIND,WATCH.
ELSE,L1.
GET,WATCH.
ENDIF,L1.
```

In the above example, if file WATCH is local to your job, the system rewinds it. (The expression FILE (WATCH,LO) returns a true value only if file WATCH is local to the job.) If WATCH is not local to the job when the IF command is processed, the system jumps to the ELSE command and then goes on to process the GET,WATCH command.

LOOPING COMMANDS

The use of iterative loops is a common programming technique. Using a loop, the programmer can write a single sequence of statements to perform multiple repetitions of a single processing task. An example of a repetitive processing task would be the reading of successive items of punched card input. Most loops use an index counter or test for a certain condition (such as the end of an input deck) to terminate looping.

NOS uses two commands, WHILE and ENDW, to form iterative loops.

WHILE AND ENDW COMMANDS

If you have a sequence of NOS commands you want to execute a number of successive times, you can form a loop around them using the WHILE and ENDW commands. The WHILE command is inserted at the beginning of the loop and contains an expression that can be evaluated to a true or false condition. The ENDW command is inserted at the end of the sequence to mark the end of the loop. The formats of the WHILE and ENDW commands are:

```
WHILE,expression,label.
ENDW,label.
```

expression and label are the same as used in the IF command. For each pair of WHILE and ENDW commands, the labels must be identical.

When the system encounters a WHILE command, the system immediately evaluates the expression contained in it. If the expression is true, the system executes all commands down to the ENDW command. The system then returns to the WHILE command and reevaluates the expression to see if it is still true. If so, the system again processes all commands down to ENDW and again returns to the WHILE command. This cycle is repeated until a false value is returned for the expression. When a false value is returned, the system immediately skips to the ENDW command, leaves the loop, and resumes processing with the next command following ENDW. This sequence is diagrammed in Figure 8-3.

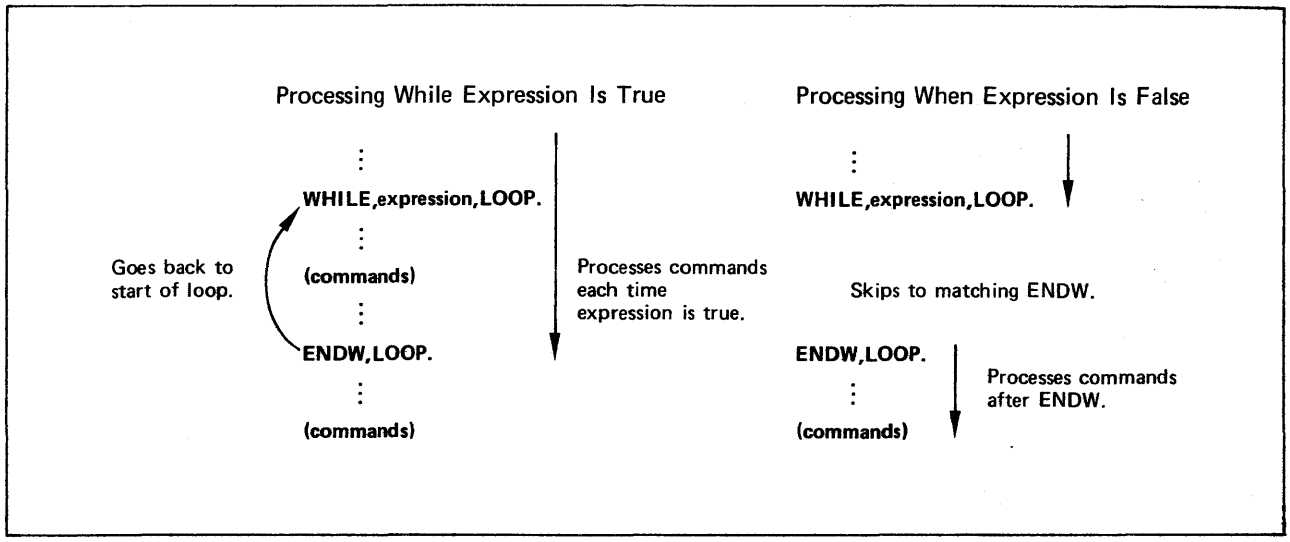


Figure 8-3. WHILE/ENDW Processing

You can use either an index counter or a change of condition to terminate looping. To terminate looping using an index counter, use one of the control registers R1, R2, R3, or R1G as described under Setting and Displaying Symbolic Name Values later in this section. To terminate looping on change of condition, you must construct the WHILE command expression such that the change of condition that is to stop looping will cause the value of the expression to become false. Figure 8-4 contains examples of iterative loops.

NOS PROCEDURE COMMANDS

A NOS procedure is a sequence of commands that performs a specific task or tasks. Figure 8-4 shows you what a procedure looks like. This file can be recognized as a procedure file because it begins with a procedure header directive (identified by the .PROC keyword) and ends with a REVERT command.

In a batch job, a procedure functions much like a program subroutine. You can pass parameter values on the procedure call in the same way that you pass values to a subroutine on a subroutine call. When a procedure finishes processing, it returns control to the calling job or procedure.

When called from an interactive job, a procedure functions more like a user-defined system command. You can enter the procedure name followed by procedure parameters in the same way you enter a system command with parameters.

The Flow Control commands pertaining to procedures are the BEGIN command and the REVERT command. The BEGIN command is used to call a procedure from a job or from another procedure. Values to be passed to a procedure are specified as BEGIN command parameters. The REVERT command terminates a procedure and returns control to the calling job or procedure.

The formats of the BEGIN command and the REVERT command are listed in Table 8-1. Both commands are described more fully in Section 9.

Procedure	Description
.PROC,COPY3*I, FILE"Name"=(*F).	Indicates that substitutions will be made for FILE.
SET,R1=0.	Assigns a value of 0 to control register 1.
WHILE,R1.LT.4,GO.	Executes the commands between this command and ENDW,GO until R1 is not less than 4.
GET,FILE.	Makes FILE a local file.
ROUTE,FILE,DC=LP.	Prints file FILE at central site line printer.
SET,R1=R1+1.	Increases the value of R1 by 1.
ENDW,GO.	Returns to the WHILE command.
REVERT,NOLIST.	Ends the procedure.

Terminal Session	Description
/get,copy3 /copy3,file=employ	The user calls procedure COPY3 on local file COPY3 and substitutes EMPLOY for FILE.
/dayfile 08.35.35.GET,COPY3. 08.36.01.COPY3,FILE=EMPLOY. 08.36.01.SET,R1=0. 08.36.01.WHILE,R1.LT.4,GO. 08.36.01.GET,EMPLOY. 08.36.01.ROUTE,EMPLOY,DC=LP. 08.36.01. ROUTE COMPLETE. JSN IS AAGO. 08.36.01.SET,R1=R1+1. 08.36.01.ENDW,GO. 08.36.02.WHILE,R1.LT.4,GO. 08.36.02.GET,EMPLOY. 08.36.02.ROUTE,EMPLOY,DC=LP. 08.36.02. ROUTE COMPLETE. JSN IS AAGP. 08.36.02.SET,R1=R1+1. 08.36.02.ENDW,GO. 08.36.02.WHILE,R1.LT.4,GO. 08.36.02.GET,EMPLOY. 08.36.02.ROUTE,EMPLOY,DC=LP. 08.36.02. ROUTE COMPLETE. JSN IS AAGQ. 08.36.03.SET,R1=R1+1. 08.36.03.ENDW,GO. 08.36.03.WHILE,R1.LT.4,GO. 08.36.03.GET,EMPLOY. 08.36.03.ROUTE,EMPLOY,DC=LP. 08.36.03. ROUTE COMPLETE. JSN IS AAGR. 08.36.03.SET,R1=R1+1. 08.36.03.ENDW,GO. 08.36.03.WHILE,R1.LT.4,GO. 08.36.03.ENDW,GO.	The user displays the dayfile to see what happened.

Figure 8-4. Using the SET Command with the WHILE and ENDW Commands

USING EXPRESSIONS IN FLOW CONTROL COMMANDS

As you have just seen, the flow control commands IF and WHILE contain conditional expressions that determine whether or not the system skips a sequence of commands. These expressions are similar in form to arithmetic and logical expressions used in mathematics, such as:

```
x<10
```

This mathematical expression has two operands, a variable (x) and a constant (10). The operands are separated by an operator (<) which defines the relationship between the operands. Notice also that this expression can be evaluated to a true or false conclusion; in any given situation, either x<10 is true or it is not true.

Expressions used in flow control commands are similar to the above example in that they are also composed of constants, variables and operators. When used in an IF or WHILE command, expressions must evaluate to a true or false conclusion since the action of the IF or WHILE command containing the expression is dependent on the result. Expressions can also appear in the SET and DISPLAY commands.

Table 8-2 shows some of the operators that can be used in NOS expressions. The arithmetic and relational operators are used with numeric operands such as the time of day or the contents of a control register. The logical operators are used with file names or symbolic names referring to logical quantities or conditions such as job origin type, file type, or device type.

Variables used in NOS expressions are of two types: symbolic names and procedure parameters. Procedure parameters are discussed in detail in the next section. Symbolic names are names which have special significance to the system. Some basic symbolic names are shown in Table 8-3. You can find the complete list in volume 3.

Table 8-2. Operators in NOS Expressions

Arithmetic Operators	
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
Relational Operators	
.EQ.	Equal to
.NE.	Not equal to
.LT.	Less than
.GT.	Greater than
.LE.	Less than or equal to
.GE.	Greater than or equal to
Logical Operators	
.OR.	Inclusive OR (Either or both operands must be true.)
.AND.	AND (Both operands must be true.)
.NOT.	Complement (The operand following .NOT. must be false.)

Table 8-3. Some Commonly Used Symbolic Names

Symbolic Name	Significance
DATE	The date in the form yymmdd.
TIME	The time of day in the form hhmm.
OT	The job origin type.
SC	The job service class.
R1	Control register 1.
R2	Control register 2.
R3	Control register 3.
RIG	Global control register 1.
EF	Error Flag.
AS	File is assigned to your job (that is, the file is a local copy of an indirect access file or is an attached direct access file).
BOI	File is positioned at beginning-of-information.
EOI	File is positioned at end-of-information.
LO	File type is local.
PM	File is an attached direct access file.
TP	File is on magnetic tape.
TT	File is assigned to a terminal.
RD	File has read permission.
WR	File has write permission.
BCO	Local batch job origin.
IAO	Interactive job origin.
RBO	Remote batch job origin.

Following are some examples to show you how operators and symbolic names are used in expressions:

Expression	Description
TIME.EQ.0945	This expression says that the current time is 9:45. (TIME is one of the symbolic names listed in Table 8-3.) The system checks the current time against 0945 and determines whether the expression is true or false.

Expression	Description
EF.GT.0	This expression says the current value for EF (Error Flag) is greater than 0. (A value greater than 0 indicates that an error has occurred.) The system checks the value of EF to determine whether the expression is true or false.
OT.EQ.BCO.OR.OT.EQ.RBO	This expression says that the job origin type (OT) is either local batch origin (BCO) or remote batch origin (RBO).
R1+R2	This expression adds the contents of control registers R1 and R2.

Many of the symbolic names have values that can be set and displayed using the SET and DISPLAY commands.

SET COMMAND

The SET command assigns values to symbolic names. Its format is:

```
SET,symbol=expression.
```

symbol can be one of the following:

Symbol	Description
R1,R2, R3	Local control registers. You can use the registers to store numbers used in the job or procedure. You can use them as counters in a WHILE, ENDW loop (an example of this follows); or you can have the system set them if a certain condition is true. The value of a local control register can be passed to a procedure and can be changed within the procedure. However, when the procedure is finished (that is, when the REVERT command is processed), the register is restored to the value it had when the procedure was called.
R1G	Global control register. This register functions in the same way as the local control registers, with one exception. R1G is not restored to a previous value at the end of a procedure. Once the value of R1G is set by a SET command, it remains unchanged unless it is modified by a subsequent SET command.
EF	Error flag. The system sets this flag when errors occur. You can also set the flag during a job or procedure. When the procedure is finished, the flag is restored to the value it had at the start.

expression can be a number or any expression that, when evaluated, is a number.

Acceptable values for the control registers (R1, R2, R3, and R1G) are integers between and including -131071 and 131071.

Acceptable values for EF are integer values from 0 to 63. NOS has a number of system-defined error flags used to report various types of error conditions. Each error flag can be referenced either by a symbolic name or by a specific octal value. For example, a CPU arithmetic error can be referenced either by the symbolic name ARE or by the value 3B. When referencing error flags in your jobs or procedures, it's best to use symbolic names, since the octal values are subject to change. Following are some examples of the SET command.

Command	Description
SET,R1=1.	Control register 1 has a value of 1.
SET,R2=1.	Control register 2 has a value of 1;
SET,R3=R2+1	Control register 3 has a value of 2 (R2+1=1+1).
SET,EF=ARE.	The error flag (EF) is set to a value which means there has been an arithmetic error (ARE).
SET,EF=0.	The error flag is set to zero; this means there are no errors.

Figure 8-4 shows a NOS procedure that sets the value of register R1 and uses R1 as a loop counter. The figure also shows the segment of the dayfile resulting from execution of the procedure.

DISPLAY COMMAND

The DISPLAY command evaluates an expression and puts the resulting number in the job dayfile. Its format is:

```
DISPLAY,expression.
```

The system evaluates expression to produce a number. The number is truncated to an integer and put in the dayfile. If your job is interactive, the results are also displayed at the terminal. The number is displayed as both decimal and octal integers. The largest decimal number that can be displayed is 10 digits; the largest octal number is 20 digits. If the value displayed is either GT-9999999999 or LT-9999999999, the decimal number has more than 10 digits. If zeros are displayed, the number is larger than $2^{48}-1$.

The following sample dayfile shows several display operations.

Dayfile	Description
08.37.06.DISPLAY,EF. 08.37.06. 0 0B	The value associated with the error flag (EF) is 0.
08.37.41.SET,EF=ARE.	The error flag is set to arithmetic error (ARE).
08.37.55.DISPLAY,EF. 08.37.55. 3 3B	The value of arithmetic error is 3.
08.38.26.SET,R1=1. 08.38.42.SET,R3=2.	Control registers 1 and 3 are set to 1 and 2, respectively.
08.39.09.DISPLAY,R1+R3. 08.39.09. 3 3B	The value of R1+R3 is 3.
08.39.33.DISPLAY,101/5. 08.39.33. 20 24B	101 divided by 5 is 20 (decimal) and 24 (octal).

Figure 8-5 shows a NOS procedure that compiles a FORTRAN program. The SET and DISPLAY commands are used in this procedure to set and display the value of symbolic name EF. Figure 8-5 also includes the segment of the dayfile resulting from execution of the procedure.

FUNCTIONS

A function is a special type of expression that returns information of a particular type. The functions we describe here are the FILE function and the NUMBER function. The FILE function is used to test for various file attributes or characteristics. The NUMBER function determines whether a character string is numeric, rather than literal.

FILE FUNCTION

The file function determines whether a file has a specified attribute. For example, you can use the FILE function to determine whether a file is local to your job, whether or not you have permission to use the file, and whether the file is positioned at BOI. If the file has the specified attribute, the file function is true; if it does not have the attribute, the file function is false. The file function's format is:

```
FILE(lfn,expression)
```

lfn is the name of the file for which the attribute is being determined.

expression can be a symbolic name or an expression that consists of operators and symbolic names. Following are some examples of file functions:

File Function	Description
FILE(WALK,BOI)	Determines whether or not file WALK is at the beginning of the file (BOI).
FILE(STOP,AS.AND.WR)	Determines whether or not file STOP is assigned to your job (AS) and (.AND.) whether you have write permission (WR) for the file. Both conditions must be true for the file function to be true.
FILE(GO,.NOT.TT)	Determines whether or not file GO is a file assigned to your terminal (TT). The file function is true either if your job is not interactive or if the file is not (.NOT.) assigned to your terminal.

You must format the file function with the separators as shown; otherwise, an error occurs.

Figure 8-6 includes an example of a FILE function.

Procedure	Description
.PROC,FORP*I, PROGRAM"name-FORTRAN program"=(*F), FORT"name used for compile file"=(*F).	Indicates that substitutions will be made for PROGRAM and FORT in the procedure.
GET,PROGRAM. FTN5,I=PROGRAM,L=0. NOEXIT.	Makes PROGRAM a local file and calls the FORTRAN compiler; specifies to continue processing even if there are errors.
SET,EF=0.	Sets the error flag to zero.
LGO.	Executes the compiled program.
IF,EF.NE.0,MISS.	Checks to see if the value of the error flag is not zero. If EF.NE.0 is true, executes the next command; otherwise, skips to ELSE,MISS.
DISPLAY,EF.	Puts the value of the error flag in the dayfile, if EF is not zero.
ELSE,MISS.	Separates the commands; those commands after ELSE are to be executed if the error flag is zero.
REPLACE,LGO=FORT.	Keeps the compiled program as an indirect access permanent file, if EF is zero.
ENDIF,MISS.	Completes the conditional processing started by the IF command.
ONEXIT.	Reenables EXIT processing.
REVERT,NOLIST.	Ends the procedure.

Terminal Session	Description
/forp,program=long,fort=short /dayfile 08.54.28. USER DAYFILE PROCESSED. 08.55.24.FORP,PROGRAM=LONG,FORT=SHORT. 08.55.24.GET,LONG. 08.55.26.FTN5,I=LONG,L=0. 08.55.26. 56000 CM STORAGE USED. 08.55.26. 0.021 CP SECONDS COMPILATION TIME. 08.55.26.NOEXIT. 08.55.26.SET,EF=0. 08.55.27.LGO. 08.55.37. STOP 08.55.37. 15700 MAXIMUM EXECUTION FL. 08.55.37. .008 CP SECONDS EXECUTION TIME. 08.55.37.IF,EF.NE.0,MISS. 08.55.37.ELSE,MISS. 08.55.37.REPLACE,LGO=SHORT. 08.55.37.ENDIF,MISS. 08.55.37.ONEXIT.	The user calls procedure FORP on local file FORP, substitutes file LONG for PROGRAM and file SHORT for FORT, and displays the dayfile.

Figure 8-5. Using the SET and DISPLAY Commands

Procedure	Description
.PROC,TAPE1*I, FILE1"-Tape file"=(*F), VSN1"-The VSN of the tape". NOEXIT.	Indicates that substitutions will be made for FILE1 and VSN1. Specifies to continue processing if errors occur.
LABEL,FILE1,VSN=VSN1,D=PE,PO=R.	Assigns a file with read permission. The file must be nine-track and recorded at 1600 cpi.
IF,FILE(FILE1,.NOT.AS),ASSIGN.	Determines whether the file is not assigned to your job (.NOT.AS). If the file is not assigned, continues with next command; otherwise, skips to ELSE,ASSIGN.
COMMENT.Tape FILE1 not accessed	Writes comment in dayfile.
ELSE,ASSIGN.	Starts skipping if IF condition is true; ends skipping if IF condition is false.
IF,FILE(FILE1,.NOT.BOI),READ.	Determines whether the file is not at the beginning of information. If not, the next command is executed.
REWIND,FILE1.	Rewinds file to BOI.
ELSE,READ.	Starts skipping if IF condition is true; ends skipping if IF condition is false.
ONEXIT.	Resumes error processing.
COPYSBF,FILE1,OUT.	Copies tape file to file OUT, formatting for printing on a line printer.
ROUTE,OUT,DC=LP.	Prints file OUT at central site printer.
ENDIF,READ.	Ends IF,READ conditional skipping.
ENDIF,ASSIGN.	Ends IF,ASSIGN conditional skipping.
REVERT,NOLIST.	Ends the procedure.

Figure 8-6. Use of the FILE Function

Number Function

The number function determines whether a character string is numeric, as opposed to literal. If the character string is numeric, the number function is true; if the string is not numeric, the number function is false. The number function's format is:

```
NUM(string)
```

string is a string of from 1 to 40 characters.

You can use the number function in a procedure to determine if a substitution that occurs is numeric, rather than literal. As an example, the procedure header directive in Figure 8-7 has a parameter that should be numeric (the number of records to be copied). The procedure first checks to see that you entered a numeric value for that parameter when you executed the procedure; if you did, it copies the number of records specified.

Procedure	Description
.PROC,RECOPY*I, FILEIN"Input"=(*F), FILEOUT"Output"=(*F), NUMBER "of records to be copied".	Indicates that substitutions will be made for FILEIN, FILEOUT, and NUMBER.
IF,.NOT.NUM(NUMBER),PASS.	Starts processing if NUMBER is not numeric; otherwise, skips to ENDIF,PASS.
REVERT.NONNUMERIC SUBSTITUTION	Ends the procedure; includes the comment in the dayfile.
ENDIF,PASS.	Starts processing if NUMBER is numeric.
GET,FILEIN.	Makes the file to be copied a local file.
COPYBR,FILEIN,FILEOUT,NUMBER.	Copies the number of records specified by NUMBER.
REPLACE,FILEOUT.	Replaces the permanent file.
REVERT.COPYBR COMPLETE	Ends the procedure; includes the comment in the dayfile.

Figure 8-7. Use of the Number Function

NOS PROCEDURES 9

The capability to write NOS procedures is a very useful feature that you can use to simplify many of your daily job processing tasks. A procedure is a sequence of NOS commands, placed in a separate record or file, that performs a specific task or sequence of tasks. A special procedure header directive identifies the file as a procedure file, and a number of other NOS directives allow you to control additional procedure processing options. Once you have created a procedure and stored it in a permanent file, the procedure is available to you at any future time. To execute the procedure, all you have to do is make the file local to your job and enter a simple, one line procedure call.

As we said in the previous section, a procedure called from a batch job or from another procedure acts much like a program subroutine. You can use the procedure call to pass parameters to the procedure. The procedure accepts any data passed to it, performs its processing functions, and returns control to the calling job or procedure.

When called from an interactive job, a procedure acts as a user-defined system command. Once you have made the procedure file local to your job, you can execute a procedure simply by entering the file name, followed by any parameters you may have defined for your procedure. When you use the interactive procedure formats, you also have some advanced interactive features available for use in your procedures. These features include interactive prompting and help text for procedures and full screen display of procedure parameter lists and menus. Interactive prompting and help features are described in this section. Full screen display features are described in the Screen Mode section of this guide.

WHY USE PROCEDURES?

Suppose you are creating a FORTRAN program. You have written the source code on a file called MYSOURC and you are now ready to compile and run it for the first time. First you might enter an FTN5 compiler command specifying MYSOURC as the input file, LIST as the output listing file, and LGO as the compiled output file. The FTN5 command looks something like this:

```
FTN5,I=MYSOURC,L=LIST,B=LGO.
```

If it compiles without error, you can then enter an LGO loader command to run the compiled program. The LGO command looks like this:

```
LGO.
```

Now, suppose the program doesn't run quite the way you want it to on the first try. You will probably enter an FSE command to edit MYSOURCE. Then you will have to rewind LIST and LGO before you recompile and rerun the program. In other words, you will enter a sequence of commands something like this:

```
FSE,MYSOURC.
REWIND,LIST,LGO.
FTN5,I=MYSOURCE,L=LIST,B=LGO.
LGO.
```

As it happens, programs don't always work on the second or third try either, so you could find yourself entering this same sequence of commands many times. Rather than manually entering each command every time you revise the program, you could use the following procedure:

```
.PROC,TESTRUN.
FSE,MYSOURC.
REWIND,LIST,LGO.
FTN5,I=MYSOURC,L=LIST,B=LGO.
LGO.
REVERT.
```

This sequence of commands is identifiable as a procedure because it begins with a .PROC directive (also called a procedure header directive) and ends with a REVERT command. The second field of a .PROC directive always specifies the name of the procedure, so in this case the procedure is called TESTRUN.

One convenient feature of NOS procedures is that you can execute a procedure from an interactive job just by entering the name of the local file that contains the procedure. You enter the file name in response to a system prompt just as you would enter any NOS command. In effect, your procedure becomes a local system command that you have created yourself. To use a procedure in this way, all you need to do is put the procedure in a local file as the first record on the file. The file can be either direct access or indirect access.

To use procedure TESTRUN as a local command, let's put the procedure in a local file named TESTRUN. Now, every time you enter the name TESTRUN in response to a system prompt, the system will automatically:

- Call FSE to edit file MYSOURC
- Rewind files LIST and LGO (when you leave the editor)
- Compile file MYSOURC
- Execute the binary output of the compiler

PROCEDURE HEADER DIRECTIVE AND REVERT COMMAND

The first line of the TESTRUN procedure is the procedure header directive, or .PROC directive for short. The .PROC directive is always the first line in a procedure file and must begin in column 1 of the file. It identifies the file to the system as a procedure file, and it specifies the name of the procedure. In its longer forms, the .PROC directive also specifies other characteristics of the procedure. The simplest form of the .PROC directive is:

```
.PROC,procname.
```

where procname is the name of the procedure.

A procedure name can be any alphanumeric name up to seven characters long, beginning with an alphabetic character. We chose the name TESTRUN for our procedure. We could just as well have chosen MYSOURC or any other valid NOS name.

We also inserted a REVERT command at the end of our procedure. The REVERT command terminates a procedure and tells the system to return control to the calling job or procedure. The REVERT command also has some error control options which we will show you a little later.

As you have just seen, using a procedure like TESTRUN can be very helpful when you are trying to work the bugs out of a new program. It is much easier to create a six line procedure that can be executed by typing in a file name, rather than to enter four separate commands every time you edit your source program.

Notice, though, that you can only use procedure TESTRUN for a source file named MYSOURC since the FSE and FTN5 commands within TESTRUN both specify MYSOURC as the input file. This means that to use TESTRUN for another source program, you would first have to modify TESTRUN to accommodate the new source file name.

So that you won't have to actually modify a procedure for each special case or environment, NOS provides a way for you to specify temporary modifications within the procedure call. These modifications are made through the use of parameter substitutions.

PARAMETER SUBSTITUTION

The term parameter substitution refers to the procedure writer's ability to define names or other character strings as procedure variables. Instead of calling them variables, though, we call them parameters.

The advantage of defining parameters (or variables) for a procedure is that when you enter a call to a procedure with defined parameters, you can specify replacement values for the parameters right on the procedure call. Then, when you enter the call, NOS will locate each occurrence of each parameter keyword in the procedure and will replace each keyword with the corresponding value specified on the call. To illustrate how this works, let's look at a modified version of the TESTRUN procedure we just created:

```
.PROC,TESTRUN,SRCFILE.
FSE,SRCFILE.
REWIND,LIST,LGO.
FTN5,I=SRCFILE,L=LIST,B=LGO.
LGO.
REVERT.
```

There are three changes in this version of TESTRUN. To the .PROC directive in line one we added a third field containing the name SRCFILE. We have also specified SRCFILE as the edit file in the FSE command and as the input file in the FTN5 command.

Placing the name SRCFILE in the third field of the .PROC directive defines SRCFILE as a parameter keyword. As we said earlier, the first two fields in any .PROC directive specify the .PROC keyword and the name of the procedure. Starting with the third field, all remaining fields specify substitutable parameters for the procedure. In other words, we can now expand our .PROC format to:

```
.PROC,procname,parameter1,parameter2,...,parametern.
```

Since we have defined SRCFILE as a parameter keyword, we can now specify on the procedure call the name of the file to be edited and compiled. To do this, we simply type in an alternate file name as the second field on the procedure call. Suppose we enter the following procedure call:

```
TESTRUN,BLUBIRD
```

The commands that NOS executes as a result of this procedure call are:

```
FSE,BLUBIRD.
REWIND,LIST,LGO.
FTN5,I=BLUBIRD,L=LIST,B=LGO.
LGO.
REVERT.
```

Notice in the above command sequence that NOS found each occurrence of the parameter keyword SRCFILE and replaced it with the parameter value BLUBIRD. We now have a procedure that will allow us to edit, compile, and execute any FORTRAN program.

If we want to further modify TESTRUN to allow name substitution for the output listing and compile files as well, we could change the .PROC directive to something like this:

```
.PROC,TESTRUN,SRCFILE,LIST,LGO.
```

Then, if we entered the procedure call:

```
TESTRUN,BLUBIRD,NEST,FLY.
```

the commands executed would be:

```
FSE,BLUBIRD.
REWIND,NEST,FLY.
FTN5,I=BLUBIRD,L=NEST,B=FLY.
FLY.
REVERT.
```

For procedures like TESTRUN that use parameters consisting of only a keyword, parameter values specified on the procedure call are associated with parameter keywords in the .PROC directive by order of appearance. The first value replaces the first keyword, the second value replaces the second keyword, and so on.

Procedure parameters can also be defined within the procedure header directive in the form:

```
keyword=value
```

For example, we could rewrite the procedure header directive for TESTRUN as:

```
.PROC,TESTRUN,IN=SRCFILE,OUT=LIST,COMP=LGO.
```

The advantage of the keyword=value format is that it allows the procedure user to enter the parameters in any order. Using keyword parameters allows NOS to identify parameters by their keyword names, rather than by their position in the procedure call.

NOS will not substitute a parameter value for a character string that appears as a part of a larger string. To be considered as a match for a parameter keyword, a character string must be immediately preceded and followed by a separator (nonalphanumeric) character. For example, look at the following procedure:

```
.PROC,EDT,F.
ATTACH,F.
FSE,F.
REVERT.
```

This simple procedure uses the parameter keyword F to represent the name of a direct access file. What happens if we enter the following procedure call?

```
EDT,XFILE.
```

The commands executed for EDT would be:

```
ATTACH,XFILE.
FSE,XFILE.
REVERT.
```

Notice that in both places where the parameter keyword F appeared between special characters (in this case, between , and .), the keyword was replaced by the parameter value XFILE. Where the character F appeared as part of a larger string (in the acronym FSE), no substitution took place.

All the procedures we have shown you so far are what we call passive or noninteractive procedures. Although they can be called from interactive jobs as well, passive procedures are used mostly in batch jobs because they cannot take advantage of the interactive parameter prompting and full screen display capabilities of NOS. If you want to read more about passive procedures, they are described fully in Volume 3, System Commands.

Since most users will want to use the advanced interactive features of NOS, we will turn our attention now to NOS interactive procedures. Before we look at interactive procedures, though, let's first talk about a couple of things that apply to either passive or interactive procedures: special substitution characters and error control within procedures.

SPECIAL SUBSTITUTION CHARACTERS

To give you greater control over the substitution of parameter values, NOS defines three special characters to control parameter substitution. These characters are the inhibit character, the concatenation character, and the literal delimiter.

INHIBIT CHARACTER (#)

The inhibit character is the pound sign (#) in the ASCII graphic character set. (The equivalent character in the CDC graphic character set is ≡).

When placed before a parameter keyword in the procedure body, the # character inhibits replacement of the keyword by any specified parameter value. For example, suppose you would like to use the following procedure to call a FORTRAN program:

```
.PROC,RUNF,I=SOURCE,L=LIST,C=LGO.
FTN5,I=I,L=LIST,B=LGO.
LGO.
REVERT.
```

If you entered the procedure call

```
RUNF,I=TXTFILE
```

your procedure would abort because it would contain the following erroneous FTN5 command:

```
FTN5,TXTFILE=TXTFILE,L=LIST,B=LGO.
```

As you can see, the value TXTFILE was substituted not only for the FTN5 input parameter value, but also for the parameter keyword. The result is an unrecognizable parameter keyword. To resolve this problem using the # character, you simply insert a # in front of the I parameter keyword, like this:

```
.PROC,RUNF,I=SOURCE,L=LIST,C=LGO.
FTN5,#I=I,L=LIST,B=LGO.
LGO.
REVERT.
```

When searching a procedure for required substitutions prior to executing a procedure, NOS also searches for any character strings preceded by a # (except those enclosed in literal delimiters which we'll explain to you shortly). If it finds one, NOS does two things:

- It strips the # character from the beginning of the string.
- It inhibits substitution for the remainder of the string (that is, it leaves the string as written, even if the string is a parameter keyword for which a substitution was specified).

Thus, if for the above procedure we enter the call:

```
RUNF,I=TXTFILE.
```

the commands executed are:

```
FTN5,I=TXTFILE,L=LIST,B=LGO.
LGO.
REVERT.
```

CONCATENATION CHARACTER (_)

The underline (_) is the concatenation character in the ASCII graphic character set. (The equivalent CDC graphic character is ↑.)

The _ character provides a way for you to specify substitutions for a substring of a character string. When NOS encounters a _ character when searching for required parameter substitutions prior to execution, NOS does three things:

- It performs any specified parameter substitutions in the normal manner.
- It removes all underlines from the body of the procedure (except those contained in a $-delimited string).
- It joins (concatenates) the character strings that were originally separated by the _ character.

Let's look at an example. Suppose you are working on three programs, MINE0A, MINE1A, and MINE2A. You want to write a procedure to compile these programs, and you want the procedure to include a 1-character parameter to distinguish between the three programs. The following procedure accomplishes this:

```
.PROC,SHOW,V.
FTN5,I=MINE_V_A,L=0,B=LGO.
LGO.
REVERT.
```

To compile and run program MINE2A, you would enter the procedure call:

```
SHOW,2.
```

The commands processed would be:

```
FTN5,I=MINE2A,L=0,B=LGO.
LGO.
REVERT.
```

In the above example, NOS replaced the keyword V with the specified value of 2, removed the _ characters from the procedure, and concatenated the three character strings that had been separated by the underlines.

LITERAL DELIMITER ($)

The special character used to define literal strings is the dollar sign ($). The literal string has two functions in NOS procedures: it allows you to specify parameter values containing nonalphanumeric characters, and it allows you to perform arithmetic comparisons of alphanumeric character strings.

Examples of parameter values containing special characters are:

```
$REC#201$
$C SMITH$
$,LGO$
```

Note, however, that NOS does allow you to specify parameter values containing a single asterisk without delimiting the string. Therefore the undelimited character string PRG*31G, for example, could be used as a parameter value.

When the system encounters a literal string within a NOS expression, the system treats the string as an integer value that can be used in arithmetic or logical operations. For an example of this usage, take a look at the following procedure called TESTPN. TESTPN initiates a FORTRAN application program called APPL. Before TESTPN calls the program, however, it first performs a validation check by comparing a user entered character string with a value specified in the procedure.

```
.PROC,TESTPN,PN.
IF,$PN$.EQ.$PRJ/2201$,L1.
ATTACH,APPL.
FTN5,I=APPL,B=0,GO.
ENDIF,L1.
REVERT.
```

This procedure requires the user to enter project number PRJ/2201. If the user enters the number correctly, TESTPN gets a direct access file called APPL and runs it as a FORTRAN program. If the project number is not entered correctly, TESTPN terminates without executing the program.

Literal strings can be up to ten characters long and can contain any alphanumeric or special characters. To include the literal delimiter itself within a literal string, it must be written as a double dollar sign ($$). For example, if you wanted to define the character string

```
$10.00
```

as a literal string, the literal string would look like this:

```
$$$10.00$
```

TERMINATING PROCEDURES

As we have seen, a NOS procedure is a sequence of NOS commands for which execution is initiated by a single procedure call. In many cases, once you initiate a procedure, the only way you know whether or not the procedure executed successfully is by the messages returned to your job's dayfile and (if you are an interactive user) to your terminal.

In this subsection, we will show you how to terminate your procedures so as to increase the usefulness of the messages returned to you. This is particularly important when you are using nested procedures (that is, when procedure A contains a call to procedure B, procedure B contains a call to procedure C, and so on). When an error occurs during the execution of a series of nested procedures, it can be difficult to determine which procedure caused the error if you do not have your procedures properly marked.

What we are going to show you now involves two commands you are already familiar with: REVERT and EXIT.

REVERT COMMAND

First of all, let's look at a another format of the REVERT command:

```
REVERT.comment
```

As with any other NOS command, you can use the entire input line following the REVERT command terminator as a comment field. It is often helpful to use the comment field to identify the procedure terminated by the REVERT command. You can also use the comment field to send diagnostic or informative messages to an interactive user.

To show you what we are talking about, let's look again at procedure TESTPN that we created earlier.

```
.PROC,TESTPN,PN.
IF,$PN$.EQ.$PRJ/2201$,L1.
ATTACH,APPL.
FTN5,I=APPL,B=0,GO.
ENDIF,L1.
REVERT.
```

Assuming that you are an interactive user, when you execute this procedure the final REVERT command will be displayed at your terminal when the procedure is completed. Now consider the effects of the following changes to TESTPN.

```
.PROC,TESTPN,PN.
IF,$PN$.EQ.$PRJ/2201$,L1.
ATTACH,APPL.
FTN5,I=APPL,B=0,GO.
REVERT.TESTPN-- PROGRAM COMPLETED
ENDIF,L1.
REVERT.TESTPN-- INVALID USER - ACCESS DENIED
```

Notice that we have added a second REVERT command which appears in line 5. Notice also that each REVERT command now has a comment appended to it. The REVERT command in line 5, which is executed only if file APPL is executed, names the procedure being executed and explicitly informs the terminal user when the program is done. The second REVERT command is executed only if the user fails to enter the correct project number. The message in this REVERT command explains why the user was denied access to the program.

You also have the option of suppressing the display of a REVERT command. In procedure TESTPN, for example, you may feel that it is unnecesary to provide the user with a PROGRAM COMPLETED message. Rather than allowing the REVERT command (which may be meaningless to the user) to be displayed by itself, you could use the following form of the REVERT command:

```
REVERT,NOLIST.
```

When you use this format, the REVERT command will not appear either at the terminal or in the dayfile.

ERROR CONTROL

In Section 7 we showed you how to use the EXIT command to maintain control of error processing within a job. When an error occurs, the system stops executing commands until it finds an EXIT command. The system resumes execution with the first command following EXIT. This allows you to insert commands following the EXIT command that perform whatever error processing you might want to do, such as saving output files, or saving the dayfile.

You can use EXIT in the same way within a procedure. Generally, it is good practice to use the following sequence of commands to terminate your procedures:

```
REVERT.comment
EXIT.
REVERT,ABORT.comment
```

Using this terminating sequence with unique comment fields, you will always be able to trace the execution sequence of a series of procedures. All procedures that execute normally will be identified in your dayfile or at your terminal by a unique REVERT command. If a procedure aborts due to an error, that procedure will be identified by a unique REVERT,ABORT command. Of course, if you also want to include other error processing functions, you can insert the appropriate commands between the EXIT command and the REVERT, ABORT command.

REVERT,ABORT is the last form of the REVERT command that we are going to show you. REVERT,ABORT does two things. First, it sets the EF symbolic name to indicate a CPU error. We aren't going to explain CPU errors here (you can read about them in Volume 3, System Commands), but you should be aware that this feature exists. Secondly, the REVERT,ABORT command itself is displayed at an interactive terminal, thus indicating to the user that the procedure was not successfully completed.

You don't have to worry about forgetting to include a REVERT command in a procedure. The system automatically appends the following sequence of commands to the end of every procedure to ensure that all procedures are properly terminated:

```
$REVERT.CCL
$EXIT.
$REVERT,ABORT.CCL
```

The prefix $ tells the system to execute system command REVERT immediately, rather than first searching for a local file or global library set by that name.

INTERACTIVE VS. NONINTERACTIVE PROCEDURES

The noninteractive procedures we have been working with in this section can be very helpful in reducing the amount of work you have to do in performing repetitive tasks. For interactive users, however, the noninteractive procedures leave a lot to be desired. For one thing, if you have forgotten the parameter specifications for one of your procedures, or if you are using a procedure written by someone else, the only way you can get information on available parameters is to list the procedure file and examine the procedure itself. For another thing, if you want to use a procedure as a way for users to initiate one of your application programs, you would probably have to provide them with all required information about the program before they log in, since there is no easy way you can give them interactive help when they are using the procedure.

INTERACTIVE PROCEDURES

A primary purpose of NOS interactive procedures is to make it easier for you to provide online assistance for interactive users of your procedures. You determine the amount of online help required. If you are writing procedures for your own future use, a short prompt description for each parameter may be all you need to refresh your memory on the function and format of each parameter. On the other hand, if your procedure will be used by others who may not be familiar with the procedure and its functions, interactive procedures let you write extensive help text for the procedure itself and for each of its parameters. This information is available to the interactive user at any time.

Another significant advantage of using interactive procedures is that you can easily control the types of values that the user can enter for each parameter. Using a simple set of symbols, called checklist patterns, you can specify validation requirements for each parameter when you define the parameter in the procedure header directive. Then, when the user enters a value for a parameter, NOS compares the value against any validation requirements you have defined. If the value violates any of your requirements, NOS discards the value and prompts the user for another one.

There are two types of interactive procedures. The first type prompts you for a parameter value which you type in following the prompt. This type of procedure is distinguished by the characters *I appended to the procedure name in the .PROC directive.

The other type of interactive procedure is the menu procedure. When the user enters a call to a menu procedure, the system displays a list of possible values for each parameter. Figure 9-1 shows you what a menu procedure display looks like. To select a value, the user simply types in the number corresponding to the desired entry. Menu procedures are indicated by the characters *M appended to the procedure name.

```
/dispose


File Routing Options
 1. Print a file.
 2. Punch a file.
 3. Plot a file.

SELECT BY NUMBER OR TYPE Q TO QUIT ?
```

Figure 9-1. Menu Procedure Display

Usually when we talk about interactive procedures, you can assume we are talking about the *I format unless we specifically refer to menu procedures.

To give you a better idea how interactive procedures work, let's take a look at one now. Figure 9-2 shows an interactive procedure that gets an indirect access file and routes it to a line printer. Please take a moment to study this procedure and its parts.

You will notice that there are three differences between procedure LEARN and the passive procedures we saw previously.

- The procedure name specification in the .PROC directive has a *I appended to it. As we said above, these characters indicate to the system that LEARN is an interactive procedure.
- The parameter definition for the GRAPH parameter contains three pieces of information: the parameter keyword, a short parameter (prompt) description, and a checklist specifying the type of value that can be entered for GRAPH.
- The procedure contains help text for the procedure and the GRAPH parameter. This information is bracketed by the procedure directives .HELP and .ENDHELP.

To show you how these changes affect the interactive system responses for procedure LEARN, Figure 9-3 contains a series of examples of user input for the procedure. The numbered callout descriptions for the examples contain cross references to callouts in Figure 9-2. By following the cross references, you can get a sense of how the procedure specifications in Figure 9-2 affect the interactive system responses in Figure 9-3.

As you can see from the examples in Figure 9-3, it is an easy matter for the user to get interactive help for a procedure. If the user enters an incorrect value or fails to enter a value for a parameter, the system prompts the user for a correct value. To get a help text description of a procedure, the user enters the procedure name followed by a question mark. To get the help description of a parameter, the user enters a question mark in response to the parameter prompt.

INTERACTIVE PROCEDURE HEADER DIRECTIVE

The procedure header directive for interactive procedures can contain up to 50 parameters. The procedure header can be continued on subsequent lines by dividing it at a valid separator character. The separator can be placed at the end of the continued line or at the beginning of the continuation line. For purposes of clarity, it's a good idea to place each parameter definition on a separate line.

Except for help text and parameter descriptions, the procedure must be typed in all upper case characters.

Aside from the *I characters appended to the procedure name, the interactive procedure header differs from the header for passive procedures only in the parameter definition format. For interactive procedures, the parameter format is:

```
keyword"description"=(checklist)
```

keyword	a 1- to 7-character name specifying the parameter keyword.
description	an optional 1- to 40-character description of the parameter.
checklist	an optional list of one or more parameter values and/or checklist patterns.

The checklist is a very important part of the parameter definition. Knowing and using the checklist options available will greatly increase the amount of control you have in specifying the types of values that can be specified for a parameter. To learn how to use checklists effectively, we need to talk about how to specify parameter values in the checklist, and how to use checklist patterns.

Specifying a list of parameter values is quite simple. All you have to do is list the acceptable values within parentheses and separated by commas. As an example, the following parameter definition requires a value of YES or NO.

```
SAVEDF"Save dayfile"=(YES,YE,Y,NO,N).
```

As you can see, this checklist has five allowable values, the abbreviated forms of YES and NO, as well as the full forms.

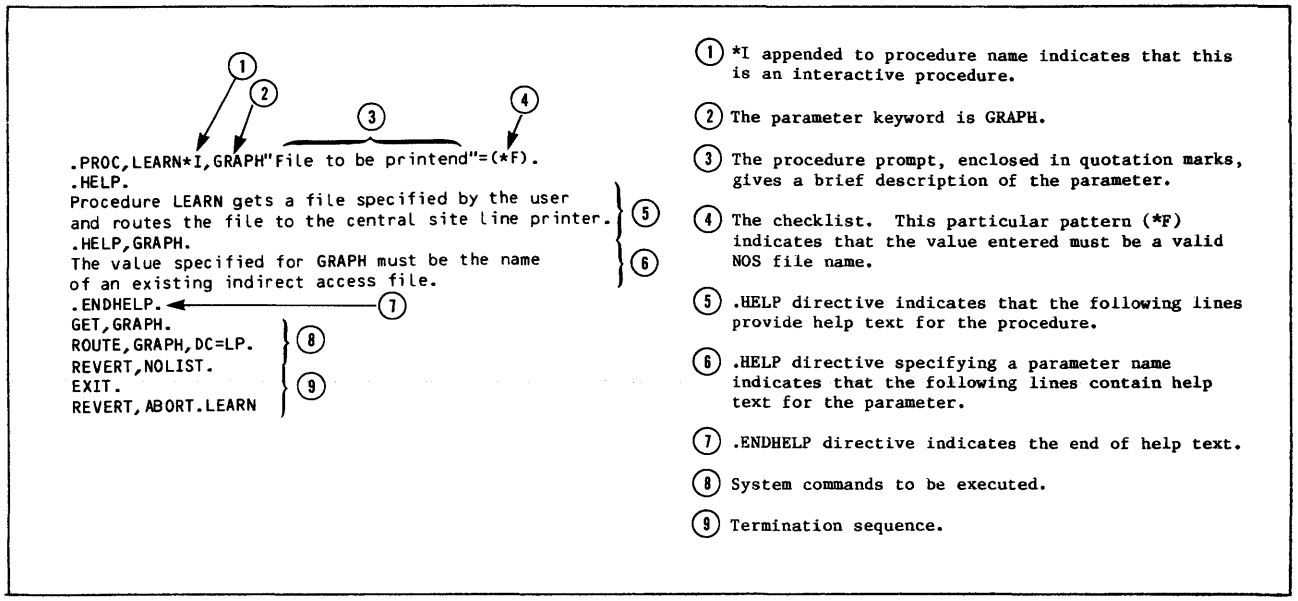

.PROC,LEARN*I,GRAPH"File to be printend"=(*F).
.HELP.
Procedure LEARN gets a file specified by the user
and routes the file to the central site line printer.
.HELP,GRAPH.
The value specified for GRAPH must be the name
of an existing indirect access file.
.ENDHELP.
GET,GRAPH.
ROUTE,GRAPH,DC=LP.
REVERT,NOLIST.
EXIT.
REVERT,ABORT.LEARN
1 *I appended to procedure name indicates that this is an interactive procedure.
2 The parameter keyword is GRAPH.
3 The procedure prompt, enclosed in quotation marks, gives a brief description of the parameter.
4 The checklist. This particular pattern (*F) indicates that the value entered must be a valid NOS file name.
5 .HELP directive indicates that the following lines provide help text for the procedure.
6 .HELP directive specifying a parameter name indicates that the following lines contain help text for the parameter.
7 .ENDHELP directive indicates the end of help text.
8 System commands to be executed.
9 Termination sequence.

Figure 9-2. Interactive Procedure File Format

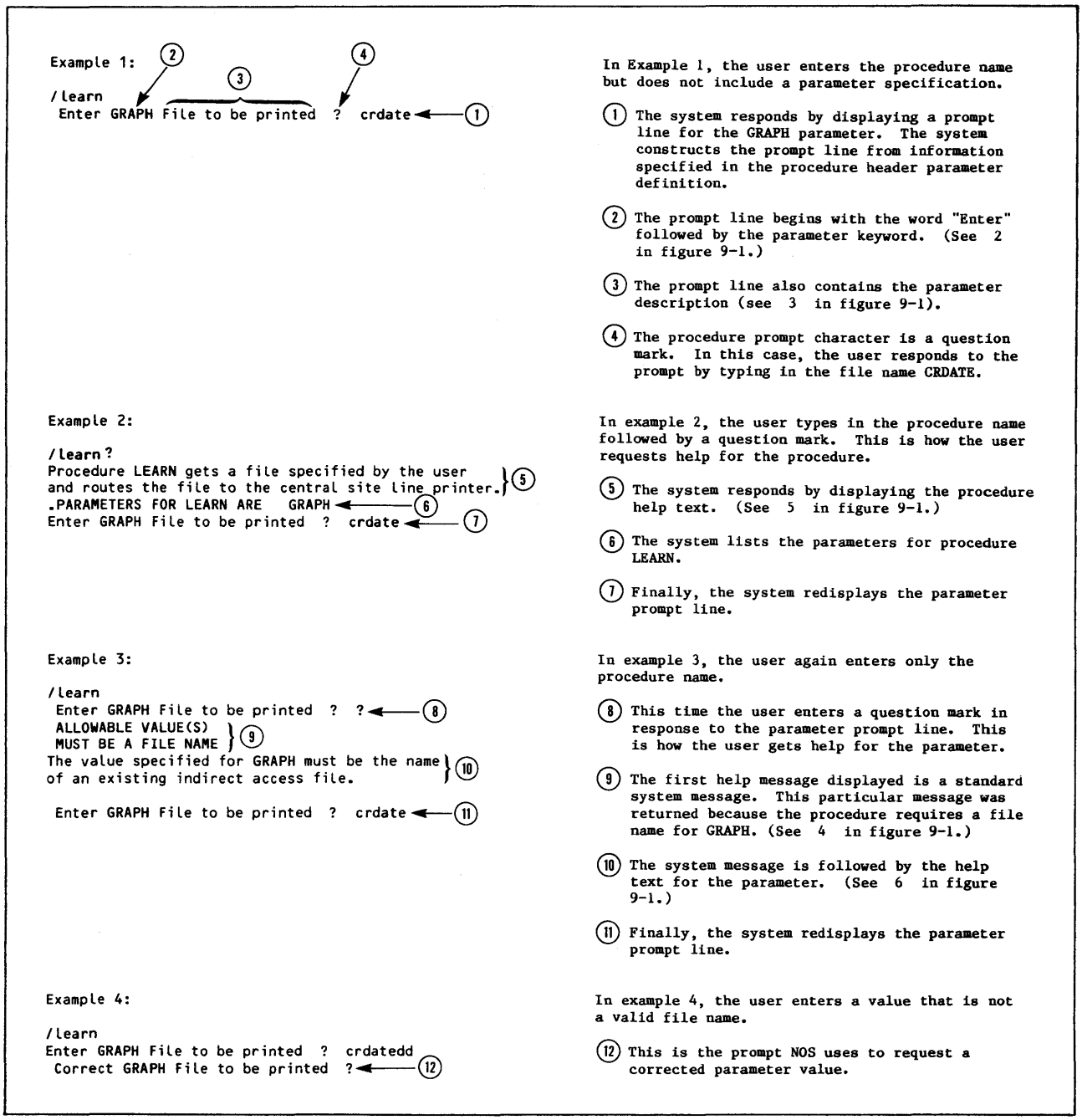

Example 1:
/learn
Enter GRAPH File to be printed ? crdate
In Example 1, the user enters the procedure name but does not include a parameter specification.
(1) The system responds by displaying a prompt line for the GRAPH parameter. The system constructs the prompt line from information specified in the procedure header parameter definition.
(2) The prompt line begins with the word "Enter" followed by the parameter keyword. (See 2 in figure 9-1.)
(3) The prompt line also contains the parameter description (see 3 in figure 9-1).
(4) The procedure prompt character is a question mark. In this case, the user responds to the prompt by typing in the file name CRDATE.
Example 2:
/learn?
Procedure LEARN gets a file specified by the user and routes the file to the central site line printer.
.PARAMETERS FOR LEARN ARE GRAPH
Enter GRAPH File to be printed ? crdate
In example 2, the user types in the procedure name followed by a question mark. This is how the user requests help for the procedure.
(5) The system responds by displaying the procedure help text. (See 5 in figure 9-1.)
(6) The system lists the parameters for procedure LEARN.
(7) Finally, the system redisplays the parameter prompt line.
Example 3:
/learn
Enter GRAPH File to be printed ? ?
ALLOWABLE VALUE(S)
MUST BE A FILE NAME
The value specified for GRAPH must be the name of an existing indirect access file.
Enter GRAPH File to be printed ? crdate
In example 3, the user again enters only the procedure name.
(8) This time the user enters a question mark in response to the parameter prompt line. This is how the user gets help for the parameter.
(9) The first help message displayed is a standard system message. This particular message was returned because the procedure requires a file name for GRAPH. (See 4 in figure 9-1.)
(10) The system message is followed by the help text for the parameter. (See 6 in figure 9-1.)
(11) Finally, the system redisplays the parameter prompt line.
Example 4:
/learn
Enter GRAPH File to be printed ? crdatedd
Correct GRAPH File to be printed ?
In example 4, the user enters a value that is not a valid file name.
(12) This is the prompt NOS uses to request a corrected parameter value.

Figure 9-3. Procedure LEARN Examples

Another checklist format allows you to specify checklist values in the form

```
string=value
```

where string is a character string entered by the user, and value is the parameter value used for keyword substitution. For example, the following parameter definition requires the user to specify version 0, 1, or 2 of a program by entering one of the character strings ZERO, ONE, or TWO:

```
VERSION"Program version"=(ZERO=0,ONE=1,TWO=2),
```

By combining simple value specifications with the string=value form, you can allow the user to enter alternate forms of the value. For example, to specify version 1 for the following parameter definition:

```
VERSION"Program version"=(0,1,2,ZERO=0,ONE=1,TWO=2),
```

the user can enter either 1 or ONE.

CHECKLIST PATTERNS

We just showed you how to define specific character strings or values that can be entered for a procedure parameter. Often, though, you will want to specify more generalized requirements for a parameter value. Probably the most frequent example of this type of parameter requirement is a parameter that accepts any valid NOS file name.

Checklist patterns provide you with an easy way to specify what restrictions or validation requirements you want to place on values that can be entered for a parameter. You have already seen a checklist pattern used in Figure 9-2. The *F pattern was used to specify that only a valid NOS file name could be entered for the GRAPH parameter.

The *F checklist pattern actually has three formats:

```
*F
*F=
*F=value
```

The *F pattern, which we saw in Figure 9-2, simply indicates that any valid file name entered as a parameter value on a procedure call will replace the parameter keyword in the procedure body.

The *F= pattern indicates that a null substitution will occur if a valid file name is specified. A null substitution essentially means that the parameter keyword is deleted (not replaced by blanks) from the procedure.

The *F=value pattern indicates that the parameter keyword will be replaced by the specified value if a file name is entered on the procedure call. For example, if *F=ROUTEF is specified in a parameter definition, any valid file name that the user enters for the parameter value causes the name ROUTEF to be substituted for the parameter keyword.

In addition to *F, NOS has four other checklist patterns: *A, *K, *N, and *Sn(set). Figure 9-4 summarizes the effects of all five patterns. You will notice that each of the patterns has three forms similar to those of *F. Notice also that the effect of the alternate forms for each pattern are like those of *F. In other words, the second form (*A=, *K=, etc.) causes a null substitution and the third form (*A=value, *K=value, etc.) uses a procedure-defined value to replace the keyword, rather than the value entered by the user on the procedure call.

Checklist Pattern	Description
*A	Allows any character string up to 40 characters. The string replaces keyword.
*A=	Any entry causes a null substitution.
*A=value	Any entry replaces keyword with value.
*F	Allows only a valid NOS file name. The file name replaces keyword.
*F=	Entry of a file name causes a null substitution.
*F=value	Entry of a file name replaces keyword with value.
*K	Allows the keyword to be entered as a value. No substitution results.
*K=	Entry of keyword causes a null substitution.
*K=value	Entry of keyword replaces keyword with value.
*N	Specifies that the parameter is optional. No substitution results.
*N=	Omission of parameter causes null substitution.
*N=value	Specifies a default value for the parameter. Omission of parameter replaces keyword with value.
*Sn(set)	Allows entry of up to n number of characters selected from set. The characters selected replace keyword.
*Sn(set)=	Entry of characters causes a null substitution.
*Sn(set)=value	Entry of characters replaces keyword with value.

Figure 9-4. Checklist Patterns

The *A checklist pattern permits the user to enter any character string up to 40 characters long for a parameter value. The string can include nonalphanumeric characters, but remember that any string that contains nonalphanumeric characters must be enclosed in literal delimiters ($). There is one exception to this rule; a string that contains a single asterisk as the only nonalphanumeric character does not have to be enclosed in delimiters.

*K allows the user to enter the parameter keyword itself as a parameter value.

*N is a very useful checklist pattern that allows you to designate a parameter as being optional. The system will not prompt the user for an optional parameter if it is omitted from a procedure call, however, the system considers all parameters to be mandatory unless they contain a *N pattern. The *N=value form of this pattern is also very useful. It allows you to specify a default value for the parameter.

*Sn(set) allows you to define a set of up to 40 characters from which the user can select up to n values. The characters in set are not separated by commas. Please note, though, that set is not a character string as such. The user can select characters from set in any order, and the order of selection determines the order of the substitution value. For example, the checklist pattern

```
*S2(ABC)
```

allows the user to select two of the three characters ABC in any order. If the user selects the characters

```
CA
```

for the parameter value, then CA is the value that replaces the keyword.

To show you how checklist patterns are used in a procedure, Figure 9-5 shows an example procedure called MEMO. MEMO is an interactive procedure that collects input and routing information on a memorandum input text file. MEMO then calls another procedure called MOPROC to put the file into memorandum format and place the reformatted file into a user-specified output file.

Let's take a closer look at the parameter definitions in lines two through seven of the procedure.

The first parameter definition in Figure 9-5 is for the memorandum input file:

```
TXTFILE"Memo Text File"=(*F,*K,*N=TXTFILE),
```

The checklist for parameter TXTFILE contains a *F to allow the user to enter any text file name and a *K to allow the use of the keyword name TXTFILE. The *N pattern defines TXTFILE as the default input file name.

The FROM parameter specifies the sender's name:

```
FROM"Name of Sender"=(*A,*N=$C. SMITH$),
```

The FROM checklist must include a *A to allow the entry of names, department numbers, or whatever. Notice that the default name is $-delimited. This is because the name C. Smith contains nonalphanumeric characters (a period and a space).

The TO checklist also contains a *A checklist to allow name entries:

```
TO"Name of Receiver"=(*A,DIST,*N=DIST),
```

The symbol for distribution (DIST) is specified as the default for parameter TO. Because DIST is also listed as a simple value in the checklist, DIST will appear as an allowable value in the parameter prompt for TO.

With the AUTOD parameter, the user specifies whether or not the automatic distribution feature is to be used:

```
AUTOD"Use Automatic Distribution List"=(YES,Y,NO,N,*N=Y),
```

The checklist specifies YES and NO values, with abbreviations, for parameter AUTOD. The default is YES.

```
.PROC,MEMO*I
,TXTFILE"Memo Text File"=(*F,*K,*N=TXTFILE)
,FROM"Name of Sender"=(*A,*N=$C. SMITH$)
,TO"Name of Receiver"=(*A,DIST,*N=DIST)
,AUTOD"Use Automatic Distribution List"=(YES,Y,NO,N,*N=Y)
,DLIST"Select Auto Distribution List(s)"=(*S4(ABCD),*N=AB)
,OUTFILE"Memo Output File"=(*F,*N=OUTPUT)
.
.HELP.
This procedure reformats a file into internal memorandum format.
.HELP,TXTFILE.
TXTFILE must be a local, indirect access, ASCII or DISPLAY code file.
.HELP,FROM.
Enter name of sender.
Default is C. Smith.
.HELP,TO.
Enter name of receiver.
.HELP,AUTOD.
Enter:  YES or Y to use the automatic distribution list.
        NO or N to include distribution list in the text file.
        Default is YES.
.HELP,DLIST.
Select one or more of the automatic distribution lists.  Make selection by
character(s):  ABCD.
.HELP,OUTFILE.
Enter the name of the file to receive the memorandum.
.ENDHELP.
GET,MOPROC.
MOPROC,TXTFILE,FROM,TO,AUTOD,DLIST,OUTFILE.
REVERT,NOLIST.
EXIT.
REVERT,ABORT.MEMO
```

Figure 9-5. Checklist Pattern Usage

The DLIST parameter lets the user choose one or more preconstructed distribution lists for the memo:

```
DLIST"Select Auto Distribution List(s)"=(*S4(ABCD),*N=AB),
```

The *S parameter allows the user to choose one, two, three, or all four of the lists, while *N defines lists A and B as the default values.

```
OUTFILE"Memo Output File"=(*F,*N=OUTPUT).
```

The output disposition parameter OUTFILE lets the user store the finished memo in a specified system file or, by default, display the file at the terminal.

PROCEDURE AND PARAMETER HELP

Writing help text for procedures and procedure parameters is very simple. All you have to do is follow the help text format shown in Figure 9-5. Help text can be written using uppercase and lowercase characters.

Help text for the procedure is optional and begins with the procedure directive

```
.HELP
```

In the lines following the .HELP directive you simply type in the help text as you want it to appear at the terminal. Any blank lines or columns you insert in the help text will appear as blank lines or columns in the terminal display. You indicate the end of the procedure help text by inserting an .ENDHELP directive or a .HELP directive for a parameter.

The .HELP directive for parameter help is optional and takes the format

```
.HELP,parametername
```

You then enter parameter help text in the same way as the procedure help.

The .ENDHELP directive signifies the end of all help text provided in the procedure. The only form of the .ENDHELP directive is

```
.ENDHELP
```

Any help text you supply must be inserted immediately following the procedure header directive as shown in Figure 9-5. The only exception is when you include screen mode directives for the procedure, in which case your help text must follow the screen mode directives. Screen mode directives are described in the Screen Mode section of this manual.

MENU PROCEDURES

Menu procedures differ from interactive (*I) procedures in two major ways:

- Each menu procedure has only one parameter.
- The user selects a parameter value from a list (menu) of displayed values. The selection is made by number; the user is not required to type in the actual value.

Menu procedures are most often used as a means of selecting and placing a call to another procedure. In other words, each menu selection typically represents a call to a procedure that performs the function or action indicated by that selection.

As an example of how this selection process works, look at the procedure called DISPOSE in Figure 9-6. The body of this procedure is divided into three sections, one for each of the options listed in the procedure header directive. Notice that each of these processing sections contains a call to a file, file MYPRINT (line 8) in the first section, file MYPUNCH (line 13) in the second section, and file MYPLOT (line 18) in the third section. Files MYPRINT, MYPUNCH, and MYPLOT each contains a procedure that performs the indicated file routing function. In effect, therefore, the purpose of menu procedure DISPOSE is to select and execute one of these three procedure files.

Figure 9-7 again shows procedure DISPOSE, this time containing help text for the procedure and for each option. Take a minute to study this figure. With the exception of the procedure header directive, the main difference between interactive (*I) and menu (*M) procedure processing is in the type of parameter substitution performed. In menu procedures, occurrences in the procedure body of the parameter keyword are simply replaced by the number of the option selected by the user. In Figure 9-7, for example, if the user selects option 1, the value 1 replaces all occurrences of the keyword OPTION. As you can see in Figure 9-7, a value of 1 entered by the user results in a TRUE evaluation for the expression in line 19. As a result, lines 20 (GET,MYPRINT.) and 21 (MYPRINT.) will be executed.

One thing that needs further explanation is the period (.) placed before IF and ENDIF in DISPOSE. .IF and .ENDIF are procedure directives. For our purposes here, you can consider that these directives function in the same way as the corresponding Flow Control commands, with one difference. Whereas the IF and ENDIF commands are used to conditionally skip intervening commands, .IF and .ENDIF effectively delete commands from the procedure before the procedure is executed.

To see the effect of .IF and .ENDIF, look at the dayfile segment in Figure 9-8. This dayfile shows the procedure commands executed as a result of a user selecting the first DISPOSE option. Since a value of 1 results in a TRUE expression in line 19 of the DISPOSE procedure, the commands (GET,MYPRINT. and MYPRINT.) bracketed by .IF,...,LABEL1 and .ENDIF,LABEL1. are not deleted from the procedure. However, since the expressions in lines 24 and 29 of Figure 9-7 are both FALSE, all commands bracketed by those .IF/.ENDIF pairs are deleted from the procedure and, therefore, do not appear in the dayfile. Notice also that the .IF and .ENDIF directives themselves do not appear in the dayfile.

Since the .IF and .ENDIF directives are much more efficient than the corresponding Flow Control commands when used in this way, we recommend that you use them when writing menu procedures. There are situations, however, in which these directives cannot replace the IF and ENDIF commands, so you should read the Volume 3 descriptions of .IF and .ENDIF before attempting to use them in other applications.

USING MENU PROCEDURES

As you can see in Figure 9-7, help text for menu procedures is very similar to interactive procedure help. As a procedure writer, you should be aware of how help text is used in menu procedures. Figure 9-9 shows a number of examples of user enteries that request help for procedure DISPOSE.

It might be helpful to try using a menu procedure at this point. Files MYPRINT, MYPUNCH, and MYPLOT are listed in Figure 9-10. If you copy these files and save them in your permanent file catalog and also make a copy of procedure DISPOSE from Figure 9-7, you can recreate the examples we showed you in Figure 9-9.

```
Line
 1        .PROC,DISPOSE*M"File Routing Options",     Procedure
 2        OPTION=                                     Header
 3        (1"Print a file.",                         Directive
 4         2"Punch a file.",
 5         3"Plot a file.").
 6        .IF,OPTION,EQ.1,LABEL1.
 7        GET,MYPRINT.                               Commands to process
 8        MYPRINT.                                   option 1 (Print)
 9        RETURN,MYPRINT.
10        .ENDIF,LABEL1.
11        .IF,OPTION,EQ.2,LABEL2.
12        GET,MYPUNCH.                               Commands to process
13        MYPUNCH.                                   option 2 (Punch)
14        RETURN,MYPUNCH.
15        .ENDIF,LABEL2.
16        .IF,OPTION.EQ.3,LABEL3.
17        GET,MYPLOT.                                Commands to process
18        MYPLOT.                                    option 3 (Plot)
19        RETURN,MYPLOT.
20        .ENDIF,LABEL3.
21        REVERT,NOLIST.
22        EXIT.                                      Terminating commands
23        REVERT,ABORT.DISPOSE
```

Figure 9-6. Procedure DISPOSE

```
Line
 1        .PROC,DISPOSE*M"File Routing Options",
 2        OPTION=
 3        (1"Print a file.",
 4         2"Punch a file.",
 5         3"Plot a file.").
 6        .HELP,NOLIST.
 7        Procedure DISPOSE routes a file to a line
 8        printer, card punch, or plotter.
 9        .HELP,1.
10        Option 1 enters a ROUTE command with the
11         DC=LP parameter.
12        .HELP,2.
13        Option 2 enters a ROUTE command with the
14         DC=PU parameter.
15        .HELP,3.
16        Option 3 enters a ROUTE Command with the
17         DC=PL parameter.
18        .ENDHELP.
19        .IF,OPTION,EQ.1,LABEL1.
20        GET,MYPRINT.
21        MYPRINT.
22        RETURN,MYPRINT.
23        .ENDIF,LABEL1.
24        .IF,OPTION,EQ.2,LABEL2.
25        GET,MYPUNCH.
26        MYPUNCH.
27        RETURN,MYPUNCH.
28        .ENDIF,LABEL2.
29        .IF,OPTION.EQ.3,LABEL3.
30        GET,MYPLOT.
31        MYPLOT.
32        RETURN,MYPLOT.
33        .ENDIF,LABEL3.
34        REVERT,NOLIST.
35        EXIT.
36        REVERT,ABORT.DISPOSE
```

(1) *M appended to the procedure name indicates a menu procedure.

(2) The menu title.

(3) The parameter keyword for this procedure is OPTION.

(4) Menu options are labeled by number. The number selected replaces the keyword in the procedure body.

(5) A description of each option that appears in the menu display.

(6) Help text for the procedure. The NOLIST parameter inhibits redisplay of the menu after procedure help is displayed.

(7) Help text for menu options.

Figure 9-7. Procedure DISPOSE (with Help)

```
list,f=dfile
09.20.18. USER DAYFILE PROCESSED.
09.20.20. USER BREAK TWO ENCOUNTERED.
09.20.25.GET,DISPOSE.
09.20.34.DISPOSE,1.
09.20.34.GET,MYPRINT.
09.20.35.MYPRINT.
09.20.42.ROUTE,CATX,DC=LP.
09.20.42 ROUTE COMPLETE.  JSN IS AASU.
09.20.42.RETURN,MYPRINT.
09.21.00.DAYFILE,L=DFILE,OP=I.
```

Figure 9-8. DISPOSE Dayfile Listing

MENU PROCEDURE HEADER DIRECTIVE

The procedure header directive for menu procedures has the format:

.PROC,procname*M,"title",keyword=(1"description$_1$",
2"description$_2$",...,n"description$_n$").

The procedure name, procname, must be a 1- to 7-character alphanumeric name beginning with an alphabetic character.

title is a 0- to 40-character menu title that appears at the top of the menu display. title can contain lowercase as well as uppercase characters.

keyword specifies the parameter keyword and can be from one to ten characters long.

description$_i$ is a 0- to 40-character string that describes the menu option. description$_i$ is displayed following the selection number in the menu display.

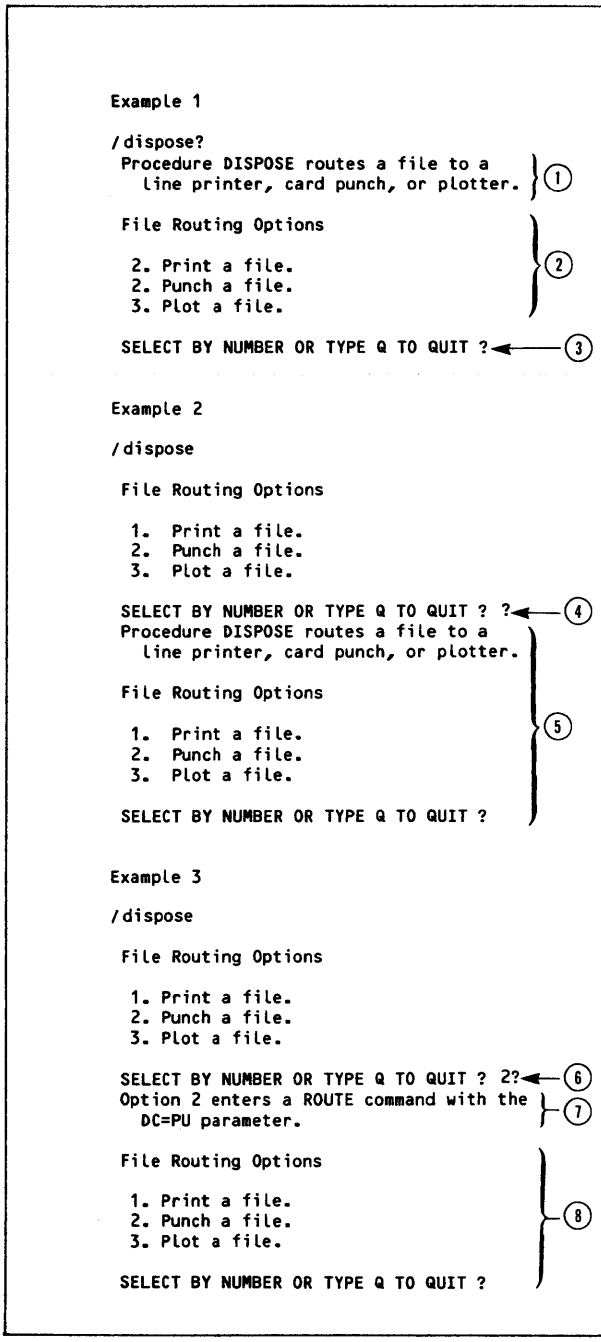
USING PROCEDURES

In this subsection we describe two ways of using procedures that can save you time and effort:

- Multi-record procedure files
- Prologue procedures

A multi-record procedure file is a file that contains a number of procedures, each contained in a separate record on the file. Putting a number of related procedures on the same file saves you the trouble of having to manipulate a separate file for each procedure.

A prologue procedure is a procedure that the system executes automatically each time you log in. You can use your prologue to set terminal characteristics or perform other job initialization tasks.


Example 1
/dispose?
Procedure DISPOSE routes a file to a line printer, card punch, or plotter.
1
File Routing Options
2. Print a file.
2. Punch a file.
3. Plot a file.
2
SELECT BY NUMBER OR TYPE Q TO QUIT ?
3
Example 2
/dispose
File Routing Options
1. Print a file.
2. Punch a file.
3. Plot a file.
SELECT BY NUMBER OR TYPE Q TO QUIT ? ?
4
Procedure DISPOSE routes a file to a line printer, card punch, or plotter.
File Routing Options
1. Print a file.
2. Punch a file.
3. Plot a file.
5
SELECT BY NUMBER OR TYPE Q TO QUIT ?
Example 3
/dispose
File Routing Options
1. Print a file.
2. Punch a file.
3. Plot a file.
SELECT BY NUMBER OR TYPE Q TO QUIT ? 2?
6
Option 2 enters a ROUTE command with the DC=PU parameter.
7
File Routing Options
1. Print a file.
2. Punch a file.
3. Plot a file.
8
SELECT BY NUMBER OR TYPE Q TO QUIT ?

In example 1, the user requests procedure help by entering the name of the menu procedure followed by a question mark. In response to the request the system prints:

(1) The procedure help text included in the procedure (see 6 in figure 9-7).

(2) The menu display listing the procedure options.

(3) A prompt to select one of the options by number.

Example 2 shows the other way of requesting procedure help. In this example, the user enters just the procedure name. The system responds by printing the menu display and selection prompt.

(4) In response to the selection prompt, the user enters a question mark to get procedure help.

(5) The system responds, as in example 1, by printing the procedure help text, followed by the menu display and selection prompt.

Example 3 shows how to get help for a new menu option. The user enters the procedure name and receives the menu display and selection prompt.

(6) In response to the selection prompt, the user enters a 2, followed by a question mark. This is how the user obtains help for option 2.

(7) The system responds by printing the help text for option 2 (see 7 in figure 9-7).

(8) The system redisplays the menu and selection prompt.

Figure 9-9. Getting Procedure Help

```
.PROC,MYPRINT*I,FN=(*F).
.HELP,FN.
Enter name of file to be printed.
.ENDHELP.
ROUTE,FN,DC=LP.
REVERT,NOLIST.
EXIT.
REVERT,ABORT.MYPRINT



.PROC,MYPUNCH*I,FN=(*F).
.HELP,FN.
Enter name of ifle to be punched.
.ENDHELP.
ROUTE,FN,DC=PU.
REVERT,NOLIST.
EXIT.
REVERT,ABORT.MYPUNCH



.PROC,MYPLOT*I,FN=(*F).
.HELP,FN.
Enter name of file to be plotted.
.ENDHELP.
ROUTE,FN,DC=LP.
REVERT,NOLIST.
EXIT.
REVERT,ABORT.MYPLOT
```

Figure 9-10. Files MYPRINT, MYPUNCH, and MYPLOT

MULTI-RECORD PROCEDURE FILES

There are two things we need to discuss about multi-record procedure files: how to create them and how to use the BEGIN command to call a procedure from a multi-record file.

Creating a Multi-Record Procedure File

There are a number of ways you can create a multi-record file. If you are using the Full Screen Editor to create a new file, you can simply insert an EOR at the end of each procedure by typing the line:

```
(EOR)
```

You can create a multi-record library file from existing files using the LIBGEN utility, which is described in Section 9, Libraries.

Another way to create a multi-record file from existing files is to use the COPYBR command to copy a number of individual files onto a single multi-record file. As you recall, COPYBR inserts an EOR at the end of each file copied.

BEGIN COMMAND

The BEGIN command calls a procedure and specifies parameter values for the procedure. Up to this point, the procedure examples in this section have used the following shortened form of the BEGIN command:

pfile,p_1,p_2,...,p_n

where pfile is the name of the file containing the procedure and p_1 through p_n are procedure parameter values. As you can see, this form of the command omits the command name BEGIN. Since the system always rewinds pfile before it executes this shortened form, you can only use this form of BEGIN to execute the first record on a procedure file.

To execute procedures other than the first procedure in a file, you must use the full form of the BEGIN command:

BEGIN,pname,pfile,p_1,p_2,...,p_n.

where pname is the name of the procedure to be executed. Using this form of BEGIN, you can execute any procedure record in a file.

Figure 9-11 shows a multi-record file containing the menu procedure DISPOSE, which we looked at earlier, as well as the secondary procedures MYPRINT, MYPUNCH, and MYPLOT. Notice that the procedure calls to MYPRINT, MYPUNCH, and MYPLOT (in lines 21, 26, and 31) must use the full BEGIN command format.

The example in Figure 9-11 demonstrates the convenience of using multi-record procedure files. With all four procedures on the same file, you do not have to retrieve four separate files from your permanent file catalog before you can execute DISPOSE. If you name the procedure file DISPOSE, you can execute the procedure just by entering:

```
GET,DISPOSE.
DISPOSE.
```

This is especially advantageous if you write procedures to be used by others. To use the DISPOSE routing procedure, for example, other users only need to know the file name, DISPOSE; they need not be aware of the existence of procedures MYPRINT, MYPUNCH, and MYPLOT.

```
Line
 1          .PROC,DISPOSE*M"File Routing Options",
 2          OPTION=
 3          (1"Print a file",
 4           2"Punch a file.",
 5           3"Plot a file.".
 6          .HELP,NOLIST.
 7          Procedure DISPOSE routes a file to a line
 8          printer, card punch, or plotter.
 9          .HELP,1.
10          Option 1 enters a ROUTE command with the
11            DC=LP parameter.
12          .HELP,2.
13          Option 2 enters a ROUTE command with the
14            DC=PU parameter.
15          .HELP,3.
16          Option 3 enters a ROUTE command with the
17            DC=PL parameter.
18          .ENDHELP.
19          .IF,OPTION,EQ.1,LABEL1.
20          GET,MYPRINT.
21          BEGIN,MYPRINT,DISP.
22          RETURN,MYPRINT.
23          .ENDIF,LABLE1.
24          .IF,OPTION,EQ.2,LABEL2.
25          GET,MYPUNCH.
26          BEGIN,MYPUNCH,DISP.
27          RETURN,MYPUNCH.
28          .ENDIF,LABEL2.
29          .IF,OPTION,EQ.3,LABEL3.
30          GET,MYPLOT.
31          BEGIN,MYPLOT,DISP.
32          RETURN,MYPLOT.
33          .ENDIF,LABEL3.
34          REVERT,NOLIST.
35          EXIT.
36          REVERT,ABORT.DISPOSE

37          (EOR)

38          .PROC,MYPRINT*I,FN=(*F).
39          .HELP,FN.
40          Enter name of file to be printed.
41          .ENDHELP.
42          ROUTE,FN,DC=LP.
43          REVERT,NOLIST.
44          EXIT.
45          REVERT,ABORT.MYPRINT

46          (EOR)

47          .PROC,MYPUNCH*I,FN=(*F).
48          .HELP,FN.
49          Enter name of file to be punched.
50          .ENDHELP.
51          ROUTE,FN,DC=PU.
52          REVERT,NOLIST.
53          EXIT.
54          REVERT,ABORT,MYPRINT

55          (EOR)

56          .PROC,MYPLOT*I,FN=(*F).
57          .HELP,FN.
58          Enter name of file to be plotted.
59          .ENDHELP.
60          ROUTE,FN,DC=LP.
61          REVERT,NOLIST.
62          EXIT.
63          REVERT,ABORT.MYPLOT
```

Figure 9-11. Multi-Record Procedure File

PROLOGUE PROCEDURES

A prologue is a procedure that executes automatically each time you log in, before you enter any system commands. In a batch job, a prologue executes after the first CHARGE command. You can create and establish a prologue for your own jobs. You might use a prologue, for example, to define your terminal's characteristics to the system or to set a default subsystem for your jobs. Creating a prologue to perform these tasks saves you the trouble of having to perform them manually each time you log in.

Your site or project administrators can also define prologues to be executed when you log in. Site or project prologues are often used to display messages of general interest to users, to set appropriate restrictions on processing options, or for various other reasons. If both site and project prologues are in effect for your user name, the site prologue is always executed first, followed by the project prologue, followed, in turn, by your own prologue if you have one.

You create your prologue as you would create any other procedure, but keep in mind that the procedure executes each time you initiate a job. Once you have created the procedure, you define it as your prologue by entering a UPROC command. The UPROC command format is:

```
UPROC,pfile.
```

where pfile is the name of the file containing the procedure.

You must also remember to save pfile as a permanent file. Otherwise, the file won't be available when the system attempts to execute it during future logins.

To change your prologue, you can either enter another UPROC command specifying a new file name, or you can simply edit your existing prologue file. To stop using your prologue, enter a UPROC command with no parameters.

The following procedure is an example of a prologue.

```
.PROC,STARTUP*I.
NOTE./NOSHOP AT YOUR SERVICE
IF,OT.EQ.IAO,OTHEROT.
TRMDEF,PG=N,PW=0.
ENDIF,OTHEROT.
ATTACH,LIB.
LIBRARY,LIB.
REVERT,NOLIST.
```

4029E

LIBRARIES 10

The NOS system uses several different types of libraries. In general, a library is a file that contains a collection of programs, routines, text files, or NOS procedures. Each program, routine, etc. is stored in a separate record on the library.

Different types of libraries are used in different ways. It is generally characteristic of libraries, however, that all information stored in a library becomes available to a job or program simply by associating the library with that job or program. For example, when you specify the name of a library in the load sequence for a program, any routines stored in the library become available for direct referencing by the program.

TYPES OF NOS LIBRARIES

There are four types of libraries. These types can be further subdivided into two classifications: libraries that you can create yourself, and libraries that are defined by the system or by your site.

User-Defined Libraries

User Library - A user library is a library created with the LIBGEN utility. To make a user library available to your job, you must first establish the library as part of a local or global library set. (We define library sets below.) User libraries can contain programs, routines, and NOS procedures to be used by your job. User libraries are edited using the LIBEDIT command.

Program Library - A program library is a source code library created and edited using either the Modify utility or the Update utility. (We discuss Modify later in this guide. For more information on Update, refer to the Update Version 1 Reference Manual.) Program libraries provide a convenient means of storing and editing source code for programs, routines, or NOS procedures. Program libraries cannot be included in a library set and cannot be referenced directly by a job or program. Information in a program must be appropriately processed (such as running it through an assembler or compiler) and placed in a user library or other local file before it can be accessed by a job or program.

Site- and System-Defined Libraries

System Libraries - System libraries contain common operating system routines available to any program executed on the system. The default system library is called SYSLIB. Routines stored in SYSLIB are always available to the loader without explicitly including SYSLIB in a library set. Other system libraries available at your site may require inclusion in a library set before they can be accessed.

User name LIBRARY - User name LIBRARY is a site-maintained set of programs, procedures, or text files available to all authorized users. User name LIBRARY is often used to disseminate information of general interest to system users. To access files on user name LIBRARY, you can issue a GET or ATTACH command specifying user name LIBRARY (that is, with a UN=LIBRARY parameter).

LIBRARY SETS

A library set is a collection of libraries associated with a job or program. Placing a library in a library set makes it possible for your job or program to directly reference NOS procedures, programs, or routines that reside on that library. There are two types of library sets, global library sets and local library sets.

A global library set is a library set that is associated with your job. Once established, the global library set remains in effect for the duration of your job, unless you explicitly release, modify, or replace it. The global library set can contain both system and user libraries. The maximum number of libraries that can be placed in the global library set at any one time depends on the number of user libraries included in the set. Table 10-1 shows the allowable combinations.

Table 10-1. Maximum Size of Global Library Set

User Libraries	Maximum System Libraries	Maximum Total Libraries
0	24	24
1	13	14
2	2	4

A local library set is a library set that is associated with a specific program load operation. Once the load operation is completed, the local library set is released. There is no restriction on the number of user libraries in a local library set.

Both the global and local library sets are established using loader commands of the CYBER Loader utility. CYBER loader operations, including the creation and use of library sets, are discussed in Section 11.

CREATING AND MAINTAINING USER LIBRARIES

The most efficient method of using and maintaining programs and procedures is to store them in a user library created by the LIBGEN utility. Whenever you change an individual program or procedure, you can create a new, updated version of the user library by using the LIBEDIT utility in conjunction with LIBGEN.

Input to LIBGEN consists of a single file containing all procedures and compiled programs or routines to be included in the user library. You can use a binary file created by a compilation as input to LIBGEN. You can also create an input file by separately copying procedures and compiled programs and routines to a single file with the COPYBR command described in section 3. Programs to be stored in a user library can be in any language.

LIBGEN use is diagrammed in figure 10-1.

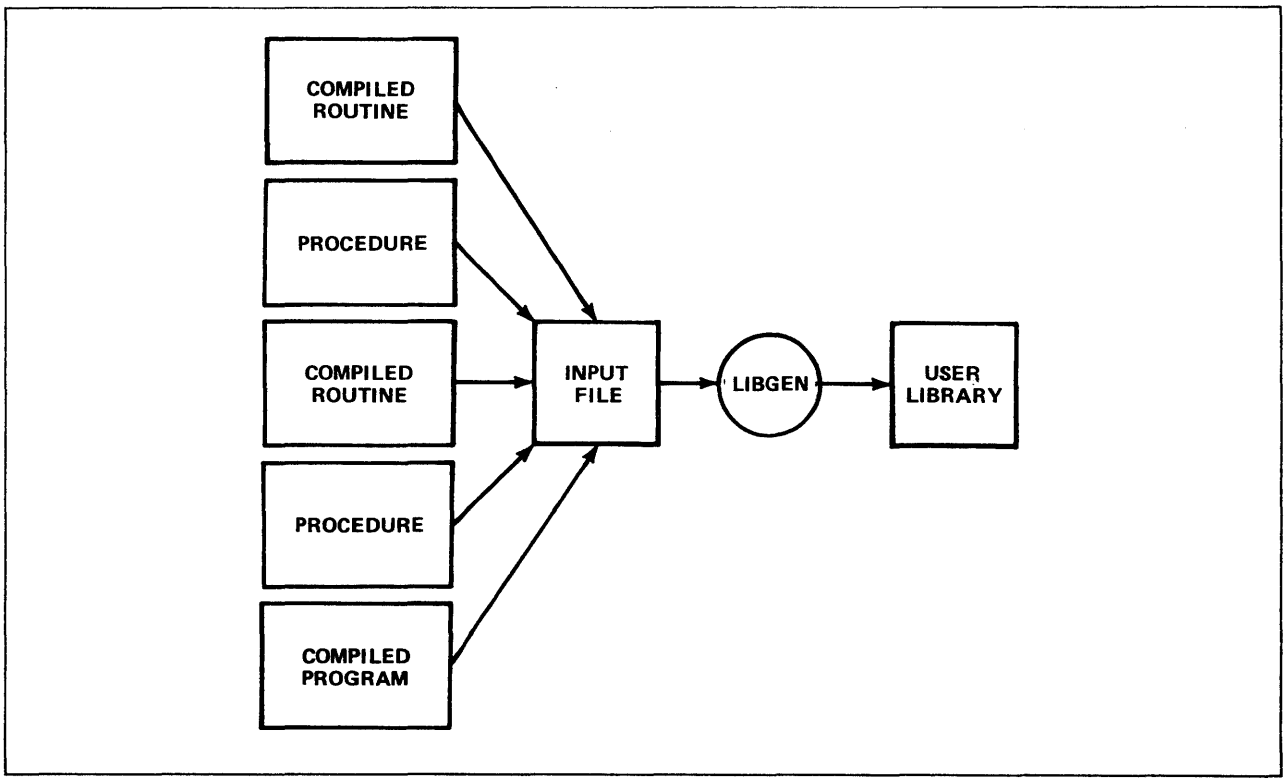


Figure 10-1. Using LIBGEN

CALLING LIBGEN

You call LIBGEN with the command:

LIBGEN,F=lfn$_1$,P=lfn$_2$,N=name.

Parameter	Description
F=lfn$_1$	The file containing absolute (ABS), overlay (OVL), procedure (PROC), relocatable (REL), or capsule (CAP) records to be made into a user library. The default is LGO.
P=lfn$_2$	The file to contain the user library. The default is ULIB.
N=name	The name of the user library being created. The default is the name specified by the P=lfn$_2$ parameter.

If you want to save the user library created by LIBGEN, use either the SAVE or DEFINE command.

CREATING A NEW USER LIBRARY

Figure 10-2 shows a sample session in which LIBGEN is used to create a user library from the binary output of an FTN5 command. File SOURCE contains source programs TEST, ADD, and SUB. The FTN5 command compiles file SOURCE to create the binary file required by LIBGEN.

The parameter B=BIN on the FTN5 command assigns the name BIN to the binary output file. This file is used as input to LIBGEN, which creates a user library named MYLIB on file LIBFIL.

The CATALOG command is used to list information about records contained in the library. The U parameter causes detailed information to be printed for each record on the library. The R parameter rewinds the library file before the catalog operation begins. Since LIBGEN does not rewind the file, the R parameter must be included to ensure that the entire file is catalogued.

Information in the catalog listing includes:

Field	Description
REC	Record number assigned by LIBGEN.
NAME	Record name. The first record is the file directory; it has the same name as the library (specified by the N parameter). Object programs incorporated as records in the library have the names assigned in the source program (program TEST, subroutine ADD, and so forth). The last record is the file index; it has the same name as the library.
TYPE	Record type. The file directory record is listed as type ULIB. The object programs are type REL, indicating that they are relocatable binary programs. The file index is listed as type OPLD.
LENGTH	Record length in octal words.
CKSUM	Number derived from the contents of the record. You can determine if any changes were made to the record by comparing the length of the CKSUM with its previous length.
DATE	Date of creation of the library.

```
/get,source=testf

/ftn5,i=source,b=bin,l=0
      0.025 CP SECONDS COMPILATION TIME.

/libgen,f=bin,p=libfil,n=mylib
  LIBRARY GENERATION COMPLETE.
/save,libfil
/catalog,libfil,u,r

          CATALOG OF LIBFIL              FILE      1
    REC   NAME      TYPE              LENGTH      CKSUM      DATE

      1   MYLIB     ULIB                  11       2675     82/05/11.
      2   TEST      REL                   72       7025     82/05/11.
           TEST
      3   ADD       REL                   34       2306     82/05/11.
           ADD
      4   SUB       REL                   35       2127     82/05/11.
           SUB
      5   MYLIB     OPLD                  11       1166     82/05/11.

      6   * EOF *        SUM =           205

CATALOG COMPLETE.
```

Figure 10-2. Creating a New User Library

CREATING AN UPDATED USER LIBRARY

You cannot modify an existing user library except by replacing it. When you change a program, you must use LIBGEN to create a new user library from a sequential file containing all the binary records to be on the library. One method of doing this is to recompile the entire source program and to input the resulting binary file to LIBGEN. A more efficient alternative, however, is to compile only the new routines and the replacement routines and to use these routines to create an updated user library.

The first step is to create a file and place in it new routines and replacement routines for those that are changed; then compile that file. You then use the compiled file along with the existing user library to create an updated user library. The following format of the LIBEDIT command creates an updated user library.

```
LIBEDIT,P=lfn1,N=lfn2,I=0,B=lfn3,U.
```

The U parameter tells LIBEDIT to call LIBGEN to create an updated user library from an existing user library. lfn_1 is the name of the existing user library file; lfn_2 is the name of the user library file being created; and lfn_3 is the name of the file that contains the compiled new and replacement routines. The I=0 parameter tells LIBEDIT that no directives are being used. Figure 10-3 shows a terminal session that adds records to a user library.

Terminal Session	Description
/get,soufile	The user makes the additional records a local file.
/get,lib1	The user makes the existing user library a local file.
/ftn5,i=soufile,b=bfile,l=0 0.007 CP SECONDS COMPILATION TIME.	The user calls the FORTRAN compiler to compile file SOUFILE, creating file BFILE.
/libedit,p=lib1,n=lib2,i=0,b=bfile,u	The user calls LIBEDIT to create updated user library file LIB2. All new routines are added at the end of the library.

```
     RECORDS WRITTEN ON FILE LIB2
         RECORD    TYPE      FILE       DATE      COMMENT

ADDED    MPY       REL       BFILE     82/05/11.
ADDED    MYLIB     OPLD      *****     82/05/11.
LIBRARY GENERATION COMPLETE.
```

Figure 10-3. Adding Records to a User Library

The updated user library contains the unchanged routines as well as the replacement and new routines. The order that the routines are in is dictated by the order of the routines on the old user library. Replacement routines assume the place of the routine being changed; all new routines are added at the end of the library.

EDITING A FILE USING LIBEDIT

You can use the LIBEDIT utility to insert, delete, and replace records on a file. The file to be updated can be created by a compilation, by the GTR utility, or by MODIFY.

The GTR utility allows you to create a multi-record file from an existing user library. This capability eliminates the need for saving the original file used to create a library. You can add, replace, and delete records within the file created by GTR using COPYL or LIBEDIT. You can then use LIBGEN to create a new user library from the updated file.

The file created by GTR is a sequential file containing records of the same format as those in the user library. Although you can use GTR to copy selected records from the existing user library, for most applications it is easier to copy all the records from the library, using GTR as a vehicle for converting a user library to a sequential file. You can use the following form of the GTR command to perform this operation.

GTR,lfn_1,lfn_2.REL/*,PROC/*,ABS/*

lfn_1 is the file containing the user library (default is OLD) and lfn_2 is the sequential format output file. This form of the GTR command copies all binary and procedure records from lfn_1 to lfn_2.

LIBEDIT creates an updated version of the input file, which you can input to LIBGEN to create a new user library. You specify detailed processing instructions to LIBEDIT through directives. Either these directives can exist on a separate file that you can create with FSE, or you can enter the directives interactively. Table 10-2 lists the most useful LIBEDIT directives. You can insert or delete records anywhere in the input file.

Table 10-2. LIBEDIT Directives

Directive Format †	Function
*TYPE,rtype	Indicates that all subsequent directives are applicable only to records of the type specified by rtype. rtype can be any valid record type including REL, ABS, PROC, OVL, CAP, or TEXT.
*BEFORE,rid,gid_1,...,gid_n	Inserts replacement records gid_1,...,gid_i immediately before existing record rid.
*AFTER,rid,gid_1,...,gid_n	Inserts replacement records gid_1,...,gid_n immediately after existing record rid.
*DELETE,rid_1,...,rid_n	Deletes existing records rid_1,...,rid_n.
*REPLACE,gid_1,gid2,...,gid_n	Replaces records gid_1,...,gid_n of the old file with records having the same names.
*LIBGEN,libname	Creates a user library named libname. This directive accomplishes the same thing as using the LIBGEN command after your LIBEDIT session. The name of the user library file created is specified by the N=lfn_2 parameter on the LIBEDIT command. This directive allows you to specify a name for the user library different from the name of the user library file. These names are the same if you use the U parameter described earlier. Also, if you use the LIBGEN directive, the original file you input must be a binary file, not a user library, as is the case with the U parameter.

† rid refers to a record in the existing file; gid refers to a record in the new file.

You call LIBEDIT with:

```
LIBEDIT,P=lfn1,N=lfn2,I=lfn3,B=lfn4,Z.directives
```

Parameter	Description
P=lfn_1	The existing file to be edited. The default is OLD.
N=lfn_2	The updated file to be created. The default is NEW.
I=lfn_3	The file containing directives. The default is INPUT. If no directives exist, you must specify I=0.
B=lfn_4	The file that contains the records to be used as additions and replacements. This file must be type REL and is usually created by a compilation. The default is LGO. If there are no records to be added, you must specify B=0.
Z	The Z parameter, when specified, tells LIBEDIT that there are directives following the command terminator. The first character after the terminator is the character that separates the directives that follow. The separator can be any character not used in the directives.

The following LIBEDIT command replaces routine SUB.

```
LIBEDIT,P=A,N=B,B=C,Z./*TYPE,REL/*REPLACE,SUB
```

The slant (/) is the separating character. The Z parameter overrides the I=lfn_3 parameter. Therefore, you cannot have an input file containing directives and also specify directives at the end of the command.

In an interactive job, you can have LIBEDIT prompt you for directives, either by using default file INPUT in the parameter I=lfn_3 or by not including the I=lfn_3 parameter when you enter the LIBEDIT command. You enter one directive after each question mark, ending with only a carriage return once you have entered all the directives. LIBEDIT then processes the directives.

Directives are not required if the only editing operation to be performed is record replacement or the addition of new records after all existing records. In this case, LIBEDIT replaces records on file lfn_1 with those records on file lfn_4 that have the same names and adds any new records to the end of the existing records.

The example in figures 10-4, 10-5, and 10-6 illustrates the process of updating a user library using GTR, LIBEDIT, and LIBGEN. In this example, the existing user library OLDLIBF contains routines TEST, ADD, SUB, and MPY. Routine TEST is to be replaced, routines ADD and SUB are to be deleted, and a new routine COMP is to be added.

Figure 10-4 shows a listing of the file containing the source statements for the new and replacement routines. Figure 10-5 shows the file containing the LIBEDIT directives required for this run. Figure 10-6 shows the terminal session for this example.

```
PROGRAM TEST
A=1.0
B=2.0
CALL MPY (A,B,C)
CALL COMP (A,B,D)
STOP
END
SUBROUTINE COMP (X,Y,Z)
Z=X**2 + Y**2
RETURN
END
```

Figure 10-4. Source Program File NEWSRC

Directive File	Description
*TYPE,REL	Indicates that the directives that follow this directive manipulate relocatable records.
*REPLACE,TEST	Replaces routine TEST.
*DELETE,ADD,SUB	Deletes routines ADD and SUB.
*AFTER,MPY,COMP	Inserts routine COMP after routine MPY.
*LIBGEN,NEWLIB	Calls the LIBGEN utility and create a user library named NEWLIB.

Figure 10-5. The LIBEDIT Directives

Terminal Session | **Description**

```
/get,oldlibf
```

The user makes existing user library OLDLIBF a local file.

```
/gtr,oldlibf,old.rel/*
 EDITING COMPLETE.
```

The user calls GTR to create sequential file OLD from the user library.

```
/ftn5,i=newsrc,b=lgo,l=0
      0.016 CP SECONDS COMPILATION TIME.
```

The user compiles file NEWSRC, which contains new and replacement records. The compiled file is named LGO.

```
/libedit,p=old,n=newlibf
ENTER DIRECTIVES -
? *type,rel
? *replace,test
? *delete,add,sub
? *after,mpy,comp
? *libgen,newlib
? carriage return
```

The user calls LIBEDIT to create new file NEWLIBF according to directives entered interactively. The file to be updated is file OLD; the file containing new and replacement routines is file LGO.

The user ends the directives. LIBEDIT updates file OLD and then calls LIBGEN to create a new user library named NEWLIB, thus creating file NEWLIBF.

```
      RECORDS WRITTEN ON FILE NEWLIBF
          RECORD     TYPE       FILE       DATE      COMMENT

 DELETED-(TEST)      REL        OLD
 INSERTED TEST       REL        LGO        82/05/11.
 DELETED-(ADD)       REL        OLD
 DELETED-(SUB)       REL        OLD
 INSERTED COMP       REL        LGO        82/05/11.
 LIBRARY GENERATION COMPLETE.
```

```
/catalog,newlibf,u,r
```

The user lists information about file NEWLIBF.

```
          CATALOG OF NEWLIBF        FILE     1
    REC   NAME       TYPE         LENGTH     CKSUM     DATE

      1   TEST       REL             72      3561  82/05/11.
           TEST
      2   MPY        REL             34      3235  82/05/11.
           MPY
      3   COMP       REL             35      5456  82/05/11.
           COMP
      4   NEWLIB     OPLD            11      4623  82/05/11.

      5   * EOF *        SUM =      174
1

 CATALOG COMPLETE.
```

```
/save,newlibf
```

The user saves file NEWLIBF.

Figure 10-6. Updating a User Library

FILE EXECUTION 11

Before you can execute a program you must do two things: compile the source file and then load it. Compiling a program translates the source file into executable machine instructions. Loading involves placing the executable machine instructions into central memory together with input/output (I/O) buffers and user/system subroutines.

PROGRAM COMMANDS

You can compile, load, and execute a program either through one command (for example, the BASIC, FTN, and FTN5 commands) or through several commands. Table 10-1 shows some of the program commands and basic parameters that are available to you. All of the commands listed can compile programs. However, the COBOL5 and COMPASS commands do not load or execute programs. The binaries parameter (B=lfn$_3$) controls the compilation of the source file [the compiled file is often referred to as the binaries of the program, because it is composed of binary (not readable) information]. The parameter defaults are in parentheses. Complete descriptions of the parameters for a specific program command are in the reference manual for the language that uses the command.

Table 11-1. BASIC, FORTRAN, COBOL, and COMPASS Commands

Program Command Name	Parameter Descriptions			
	Input File	Output File	Binaries	Execute
BASIC	I=lfn$_1$ (INPUT)	L=lfn2 (OUTPUT)†	B=lfn3 (no file)††	GO (binaries loaded and executed)
FTN	I=lfn$_1$ (INPUT)	L=lfn2 (OUTPUT)	B=lfn3 (LGO)	GO (binaries not loaded and executed)
FTN5	I=lfn$_1$ (INPUT)	L=lfn2 (OUTPUT)	B=lfn3 (LGO)	GO (binaries not loaded and executed)
COBOL5	I=lfn$_1$ (INPUT)	L=lfn2 (OUTPUT)	B=lfn3 (LGO)	†††
COMPASS	I=lfn$_1$ (INPUT)	L=lfn2 (OUTPUT)	B=lfn3 (LGO)	†††

†The L parameter controls compilation output. The execution output is determined by the K=lfn parameter.
††By default, no compile file is created for the user.
†††These commands only compile files; they do not load or execute binaries.

To avoid the necessity of recompiling a program each time you want to execute it, you can save it with the SAVE command. For example, to save a compiled FORTRAN program you could enter:

```
GET,SAD.
FTN5,I=SAD,B=COMSAD,L=0.
SAVE,COMSAD.
```

File SAD is the FORTRAN program. File COMSAD is created by the compiler and contains the compiled program; it becomes an indirect access permanent file when you enter the SAVE command.

You may also want to save a compiled program or routine so that you can use it with other programs or routines. You can group your compiled programs and routines in a user library.

LOADING PROGRAMS

Having compiled a program, you must place it in memory before you can execute it. To place it in memory, you use a system utility called the CYBER Loader.

Input to the loader consists of compiled programs and commands that specify various loader options. Loader output consists of an executable program and, optionally, a load map. The loader function is diagrammed in figure 10-1. You can direct the loader either to write the executable program to a disk file or to enter the program into memory and begin execution.

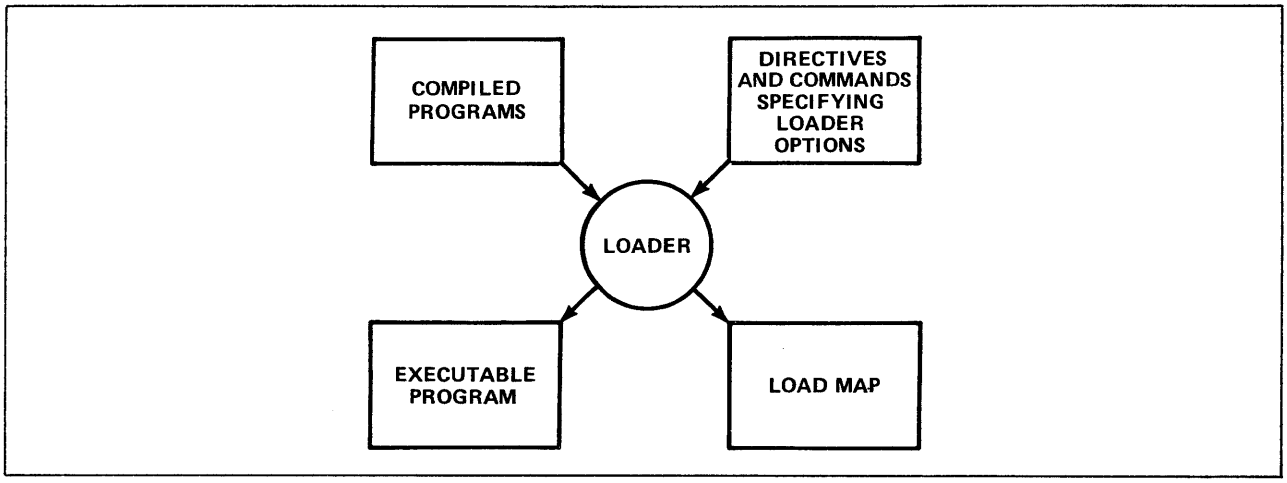


Figure 11-1. Loader Input and Output

When using BASIC or FORTRAN from an interactive terminal, you can run your programs under the BASIC or FORTRAN subsystem rather than entering the BASIC or FORTRAN command. (From the terminal, entering BASIC puts you in the BASIC subsystem and entering X,BASIC calls the BASIC compiler.) Loading as we describe it in this section does not apply to programs run under the BASIC or FORTRAN subsystem. Refer to the appropriate reference manual and to Volume 1, Introduction to Interactive Usage for more information.

The simplest type of loading occurs when you include the GO parameter on the program command. The GO parameter causes the compiler to automatically call the loader after compilation, and the loader loads and executes the binaries. You can use this form only on the BASIC and FORTRAN commands. For all languages shown in table 10-1, you can also execute the binaries by using the name call command (described later). For all languages except BASIC, the default compiled program name is LGO.

The loader has optional features, which:

- Reduce execution time field length requirements by allowing you to divide large programs into smaller units that need not be in memory at the same time.
- Increase program modularity by accessing routines stored on user or system libraries.
- Save the executable loaded program, so you can load the program once to execute it repeatedly.
- Provide information useful for program debugging and analysis by producing a load map.

Which loader options to use is a decision that you should base on the requirements of the program. A particular point to consider is that options that reduce field length requirements can increase execution time, because more time is required to move programs into and out of central memory.

TYPES OF LOADING

Following are some of the types of loading that are available to you.

- Basic loading is accomplished by commands through which all the routines required for execution are loaded into central memory at the same time. This is the most common type of loading, because most programs are not large enough to need more sophisticated techniques.
- Overlay loading is accomplished by OVERLAY directives you place in the source program. Programs are divided into smaller units called overlays. Overlays are loaded at execution time as directed by calls to the OVERLAY subroutine, which you also insert in the source program.
- Segment loading is accomplished by SEGLOAD commands in conjunction with segment directives. Programs are divided into smaller units called segments, which are organized into treelike structures. At execution time, when a segment not already in memory is required, it is loaded automatically. Segment loading requires no changes to your source program, however, you cannot have segments in your user library.
- Capsule loading is accomplished by OVERLAY and OVCAP directives. Programs are divided into smaller units, called capsules, which are loaded and unloaded as needed during execution by special subroutine calls that you include in the source program. The program's field length is adjusted when a capsule is loaded or unloaded.

Overlays, segments, and capsules each provide a method of reducing a program's field length requirements; each method has advantages and disadvantages. If you think your program may get to a size that you will want to use overlays, segments, or capsules, you should structure your original program with this in mind. Refer to the CYBER Loader Version 1 User's Guide for more information.

LOADING PROCESSES

In addition to placing compiled files into memory, loading includes relocation and satisfaction of external references.

The object code produced by a compilation is in relocatable format; that is, the addresses of variables and instructions are only temporary and are defined relative to the beginning of the program unit or of a common block. Before you can execute the program, the addresses must be made absolute, that is, defined relative to the beginning of the program's field length. The process of calculating absolute addresses is called relocation and is performed by the loader. Relocation is performed in different ways, depending on the type of loading. For overlay, segment, and absolute loading, relocation is performed only once, reducing the time required for subsequent loads.

External references can be generated by your program or by the compiler. In a program, external references include calls to subroutines and references to functions. Compiler-generated references include references to input and output statements and references to routines, such as exponentiation routines. For each external reference it encounters, the compiler generates a code sequence, called a calling sequence, which includes a branch to the referenced routine. The compiler, however, does not know the addresses of the referenced entry points. It is the function of the loader to determine the absolute addresses of the referenced entry points and to insert the addresses in the calling sequences of the referencing routines. This is the process of satisfying external references.

The loader satisfies external references in a file by matching them with entry points within other programs on the same file. If unsatisfied external references remain after this process, the loader attempts to satisfy them by searching available libraries until a matching entry point is found. When a duplicate entry point exists on the same file, or on different files, the loader follows an established search order to determine which entry point to use to satisfy the external reference. We describe the process of satisfying external references in greater detail later.

LOADING SEQUENCES

A loading sequence is one or more loader commands that define your loading operation. You must specify a loading sequence for any type of loading.

A loading operation may require one or more of the following commands.

Command	Description
LOAD	Specifies files to be loaded.
SLOAD	Specifies program units to be loaded.
LDSET	Specifies a local library set to be used in satisfying externals.

You must end each loading sequence with a completion command. The following are completion commands.

Command	Description
EXECUTE	Initiates loading and program execution.
NOGO	Initiates loading, but instead of initiating program execution, it writes the absolute program on a specified file.

You can also use the name call command (for example, LGO) to end a loading sequence, or you can specify it alone as the simplest type of loading sequence. In either case, the name call command loads the program named on it and starts execution.

When the system encounters the first loader command, it calls the loader. The loader then reads all the commands in the sequence until it encounters a completion command. The completion command causes the loader to return to the first command in the sequence. The loader then:

- Places in central memory the programs indicated on the loader commands, in the order that the commands are listed.
- Searches the libraries to satisfy external references.
- Determines the execution field length.
- Writes the load map.
- Initiates program execution or writes the program to the specified file (NOGO).

BASIC LOADING

Basic loading is loading in which all the object code required for execution is present in memory at the same time; you do not divide the object code into smaller units as in overlay, capsule, or segment loading. You accomplish basic loading entirely with commands; you need no special loader directives or subroutine calls.

The loading that most programmers are familiar with, that performed by the name call command (such as LGO), is one type of basic loading. When you enter the name call command following a compilation, the binary program in the file named is loaded into memory and execution begins. Other more complicated types of basic loading allow you to load selected program units from any number of files or from libraries. These types require loading sequences containing additional loader commands.

NAME CALL LOADING

Name call loading is the simplest type of loading. It is initiated by a name call command having the basic format:

```
name.
```

name is the name of the binary file to be loaded. The name call command places into memory the program units on the specified file, searches libraries to satisfy external references, and initiates program execution. You can separate the program units by EORs; records are loaded until the first EOF, EOI, or double EOR is reached. The file to be loaded must contain only one main program. If you specify on the name call command a file that is not executable, you get the message:

```
LAST FILE ACCESSED-filename
```

LGO is the most common name used for name call loading. (LGO is the default name of the binary file created by FORTRAN, COBOL, and COMPASS compilations.)

LOADING PROGRAMS FROM MULTIPLE FILES

You can load programs from several files into a single executable module with the LOAD command. This command's format is:

```
LOAD,file1, . . . ,filen.
```

$file_i$ is a file containing program units to be loaded. The LOAD command loads all of the routines from each of the specified files into your program's field length. Loading stops when EOR or EOF is encountered. The file cannot be a library file. Unlike the name call command, the LOAD command is not complete in itself. It requires a completion command to initiate loading and, when applicable, to initiate program execution. For example:

```
LOAD,LGO.
EXECUTE.
```

is equivalent to the name call:

```
LGO.
```

Both sequences load program units from file LGO, satisfy external references, and initiate execution.

You can load selected program units from specified files with the SLOAD command. This command's format is:

SLOAD,file,$name_1$, . . . ,$name_n$.

file is the file from which program units are to be loaded, and $name_i$ is a program unit to be loaded. As with LOAD, the specified file must be a sequential format binary file. SLOAD also requires a completion command.

In name call loading, all program units in the program must be on the same file. However, the LOAD and SLOAD commands allow you to write a program using other programs that exist on different files; the program units are merged into a single executable unit during loading.

For example, if program unit MAIN calls a subroutine SUBA, name call loading requires that the compiled versions of MAIN and SUBA exist on the same file; however, if MAIN and SUBA are on separate files, you can load them into a single executable program with LOAD and SLOAD commands.

Before executing a program, you must be sure that all needed program units that are not in libraries are explicitly loaded by LOAD, SLOAD, and name call commands and that only one main program is loaded. You can include any number of SLOAD or LOAD commands in a loading sequence; the sequence must end with a completion command (EXECUTE, NOGO, or name call).

Following are some examples of typical loading sequences.

Example 1:

The loading sequence:

```
LOAD,ABLE,BAKER,CHARLIE.
EXECUTE.
```

loads all program units on files ABLE, BAKER, and CHARLIE and initiates execution.

Example 2:

The loading sequence:

```
SLOAD,BB,MAIN,SUB1.
SLOAD,CC,SUB2,SUB3.
LOAD,DD.
BINF.
```

loads program units MAIN and SUB1 from file BB, loads program units SUB2 and SUB3 from file CC, loads the entire contents of file DD, loads the program units on file BINF, and initiates execution.

TERMINATING A LOADING SEQUENCE

You must end every loading sequence with a completion command. These commands initiate loading and either initiate execution of the program or write the program to an indicated file. The completion commands are:

- name call.
- EXECUTE.
- NOGO.

When used as a completion command, the name call command loads the contents of the file named on it and initiates execution of the resulting program. For example, the sequence:

```
LOAD,F1,F2.
LGO.
```

loads object programs from files F1, F2, and LGO, and initiates execution.

You can use the name call command to execute binary files, procedures (described in section 7), and libraries that are on a global library set local to your job (described later).

The EXECUTE command performs the completion functions and initiates execution of the resulting absolute program. This command's format is:

```
EXECUTE.
```

Unlike the name call command, you can use EXECUTE only to terminate a loading sequence; you cannot specify it by itself.

The processes of searching libraries and generating absolute addresses can be time-consuming. The NOGO command is a means of reducing the amount of loading time required for a program by saving the loaded program on a separate file. This command's format is:

```
NOGO,file.
```

The NOGO command performs the same functions as the EXECUTE command, except that NOGO does not initiate execution. Upon completion of loading, the absolute program is written to the specified file (the default file name is ABS). You can subsequently load this file into memory (usually by a name call command) and execute it; however, the time required to load this absolute file is much less than that required for a relocatable file. The NOGO command is especially useful for programs that are to be executed many times. For example, the loading sequence:

```
LOAD,PROG.
NOGO,SAVEF.
```

loads programs on file PROG and writes the resulting absolute program to file SAVEF. You can subsequently execute the saved program with the name call:

```
SAVEF.
```

ENTERING LOADING SEQUENCES FROM THE TERMINAL

The loader processes a loading sequence entered from an interactive terminal as a single unit rather than as individual commands. Since it expects the entire loading sequence to be entered together, do not enter commands other than loader commands once you have started the sequence.

The rules for entering a loading sequence interactively are essentially the same as for specifying a loading sequence in a batch job. The sequence is initiated by the first loader command and is ended by a completion command. When you enter the first loader command, the loader reads (but does not execute) the command and prompts you for the next loader command. The loader prompt looks like this:

```
LDR>?
```

You can then enter the next command. The loader continues to prompt you until you enter a completion command, at which time processing begins.

LOADING PROGRAMS FROM LIBRARIES

You can load and execute binary programs from user libraries and from system libraries.

System libraries contain system interface routines required by most other programs. The set of system libraries that exists on a system is an installation option.

User libraries are libraries you create from sequential files of compiled programs or routines using the LIBGEN utility.

Unlike program units on files, you do not load library routines explicitly; the loader searches libraries and loads the required routines in the process of satisfying external references. The loader satisfies as many external references as possible with routines you have already loaded into memory using LOAD and SLOAD commands. If any unsatisfied external references remain, the loader searches for the referenced routines on the libraries that are available to it. You must make sure that the needed libraries are available to the loader before you initiate loading.

The default system library, SYSLIB, is always available to the loader. If you want to use any other system library, either you or the compiler must make it available (for information, refer to the reference manual for the language you are using).

You can make user libraries available to the loader either by putting them in a global library set with the LIBRARY command or by putting them in a local library set with the loader command LDSET.

The LIBRARY command's format is:

```
LIBRARY,file1, . . . ,filen/p.
```

$file_i$ is a file containing a user library, and p is one of the following.

p	Description
A	Adds the libraries specified by $file_i$ to the global library set.
D	Deletes the libraries specified by $file_i$ from the global library set.
R	Replaces the global library set with those libraries specified by $file_i$. R is the default.

The maximum number of libraries that can be contained in the global library set is 2 user libraries and 2 system libraries, 1 user library and 13 system libraries, or 0 user libraries and 24 system libraries. You can clear the global library set by entering a LIBRARY command with no parameters.

```
LIBRARY.
```

After you enter this command, the only libraries available to the loader are the default system library and libraries declared in LDSET commands.

The following example illustrates using LIBRARY commands in name call loading. This example assumes the existence of user libraries ALIB, BLIB, and CLIB and of binary files LG01, LG02, and LG03.

```
LIBRARY,ALIB,BLIB.
LG01.
LIBRARY,CLIB.
LG02.
LIBRARY.
LG03.
```

The name call LG01 searches libraries ALIB and BLIB and the default system library SYSLIB to satisfy external references. The name call LG02 searches libraries CLIB and SYSLIB. The name call LG03 searches only SYSLIB.

Within a loading sequence, you can make libraries available by using the LDSET command. The LDSET command's format is:

```
LDSET,LIB=file1/ . . . /filen.
```

$file_i$ is the local file name of a file containing a user library. Since LDSET is a loader command, you can only use it within a loading sequence. Libraries specified in LDSET commands compose the local.library set and, as such, are available to the loader only during the loading sequence in which they appear. You can include any number of LDSET commands in a loading sequence; all libraries specified in the commands become part of the local library set. LDSET commands can appear anywhere within a loading sequence. The libraries are searched in the order of their appearance in the loading sequence. For example, in the loading sequence:

```
LDSET,LIB=LIB1/LIB2.
LOAD,LGO.
LOAD,BIN.
EXECUTE.
```

binary programs from LGO and BIN are placed in memory; libraries LIB1, LIB2, and SYSLIB are searched; and execution is initiated, in that order.

LIBRARY SEARCH ORDER

In the process of satisfying external references or locating an entry point for a name call command, ambiguity can arise when the same entry point or program name occurs more than once within libraries available to the loader. For this reason, a rigorous search order exists, so that you can always determine how the loader satisfies each external reference.

The loader search order for external references is:

- Programs already loaded.
- Global library set (includes libraries specified in LIBRARY commands).
- Local library set (includes libraries specified in LDSET commands).
- SYSLIB (system default library).

The loader search order for the name call command is:

- Local files.
- Global library set.
- Local library set.
- SYSLIB.

Within each library set, the loader searches libraries in the order that you included them in the set.

If the loading of programs from libraries produces new external references that must be satisfied from the library set, the loader satisfies these references when it first encounters the appropriate entry point or program during the search. If it encounters the end of all the library sets and unsatisfied references remain, it searches again from the beginning. This circular search continues until either it satisfies all references or it has searched the entire library set once with no new satisfaction of references.

If unsatisfied external references exist at the completion of loading, the system displays an informative message. A fatal execution error results, if a statement containing an unsatisfied external reference is executed.

Figure 11-2 contains two examples of the library search order for external references. These examples show terminal sessions in which the user enters a LIBRARY command and a loading sequence. In the first example, the global library set consists of ALIB and BLIB. No user libraries are explicitly assigned to the local library set; however, if any system libraries are required by the program, they will automatically be included in the local library set. In the second example, the global library set consists of ULIB, and the local library set consists of ZLIB and any system libraries required. In both examples, the default system library is SYSLIB.

EXAMPLE 1

Terminal Session:

```
/get,comp1,comp2
/library,alib,blib
/load,comp1,comp2
LDR>?execute

     .004 CP SECONDS EXECUTION TIME.
```

Library Search Order:

```
ALIB

BLIB

SYSLIB
```

EXAMPLE 2

Terminal Session:

```
/get,comp3
/library,ulib
/ldset,lib=zlib
LDR>?load,comp3
LDR>?execute

     .005 CP SECONDS EXECUTION TIME.
```

Library Search Order:

```
ULIB

ZLIB

SYSLIB
```

Figure 11-2. Library Search Order

ADDITIONAL LOADER OPTIONS

Additional commands allow you to specify various loader options. One of the most useful loader commands is the LDSET command. This command controls a number of loader options, including establishing a local library set (described earlier), presetting unused memory, selectively loading or omitting routines, setting the default rewind indicator, and controlling the load map. These capabilities are described in the CYBER Loader Version 1 User's Guide.

7422G

SCREEN MODE 12

NOS screen mode is a feature that allows you to make use of full screen displays for entering input to programs, creating displays, editing files using FSE, and running procedures. The screen is the basic unit of interaction, rather than the line. NOS screen mode can support most character mode asynchronous terminal models. Consult your site administration for information on screen mode support for your particular terminal.

Before you can use screen mode, you must enter the SCREEN command. This command puts your terminal in screen mode.

If you are using the Viking 721, for example, the format of the SCREEN command is:

```
SCREEN,721.
```

You can return to line mode by entering the LINE command. The format for the LINE command is:

```
LINE,721.
```

The displays on the terminal screen can have different formats. For some system commands and utilities, the system defines the format of the screen. Such is the case with the Full Screen Editor (FSE). For procedures, the system also defines the format of the screen, but you control the text of the prompts that appear. For screen mode input to application programs written in FORTRAN, COBOL, or Pascal, you must define the format of the screen. The NOS Screen Formatting Reference Manual describes in detail how you format the screen.

The terminals that can be used in screen mode have programmable function keys. The location of the keys on the keyboard varies depending on the terminal. On the Viking 721, the keys are located across the top and down the right side of the keyboard.

The function of each key varies with the application you are using. The system defines the function of six of these programmable function keys when using procedures. In FSE, the function keys have system-defined defaults, but you can redefine their functions. For application programs that use screen mode input, you must define the function of any key you want to be operable. In all these cases, you can change the labels of keys, which appear at the bottom of the screen. The NOS Screen Formatting Reference Manual and the NOS Full Screen Editor User's Guide describe how to make these changes.

The rest of this section shows you how to use FSE and how to run procedures while in screen mode.

USING THE FULL SCREEN EDITOR IN SCREEN MODE

The Full Screen Editor is a text editor enabling you to create and change your files. You can use FSE either as a screen editor or as a line editor. In this section we will be editing files using FSE as a screen editor. We use the Viking 721 for all examples and explanations of keys.

When screen editing, your file is displayed page by page enabling you to edit it more easily. In this section we discuss only the basic FSE functions needed to create and edit a file. Later in this section, we will discuss how to get online help concerning these functions. For complete information on FSE, refer to the Full Screen Editor User's Guide.

Figure 12-1 is the Viking 721 terminal keyboard and the keys used in this section.

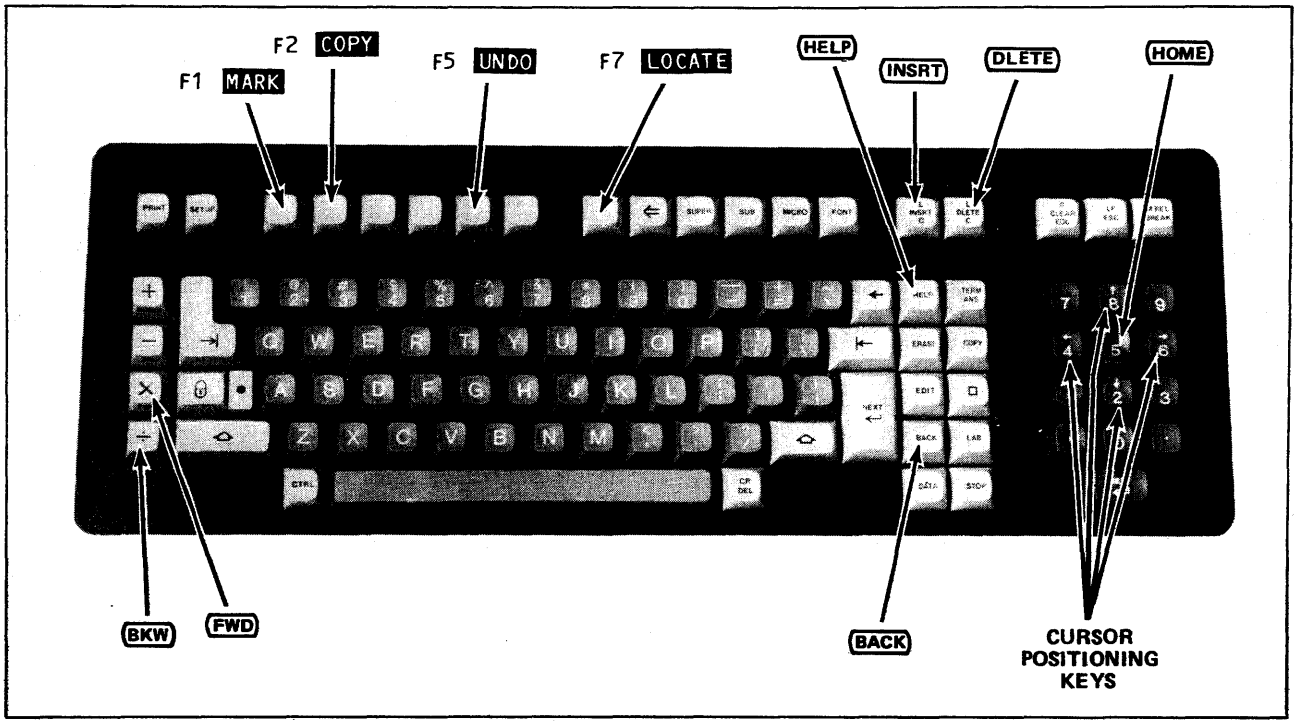


Figure 12-1. Viking 721 Terminal Keyboard

STARTING FSE

To start FSE, enter the following command.

```
FSE,filename.
```

filename is the name of a local file to be edited. If the specified file is not local, FSE creates a file with that name.

CREATING FILES

To create a local file, enter the name of the file on the FSE command. For example, to create local file BIKE, enter:

```
FSE,BIKE.
```

The screen in figure 12-2 appears.

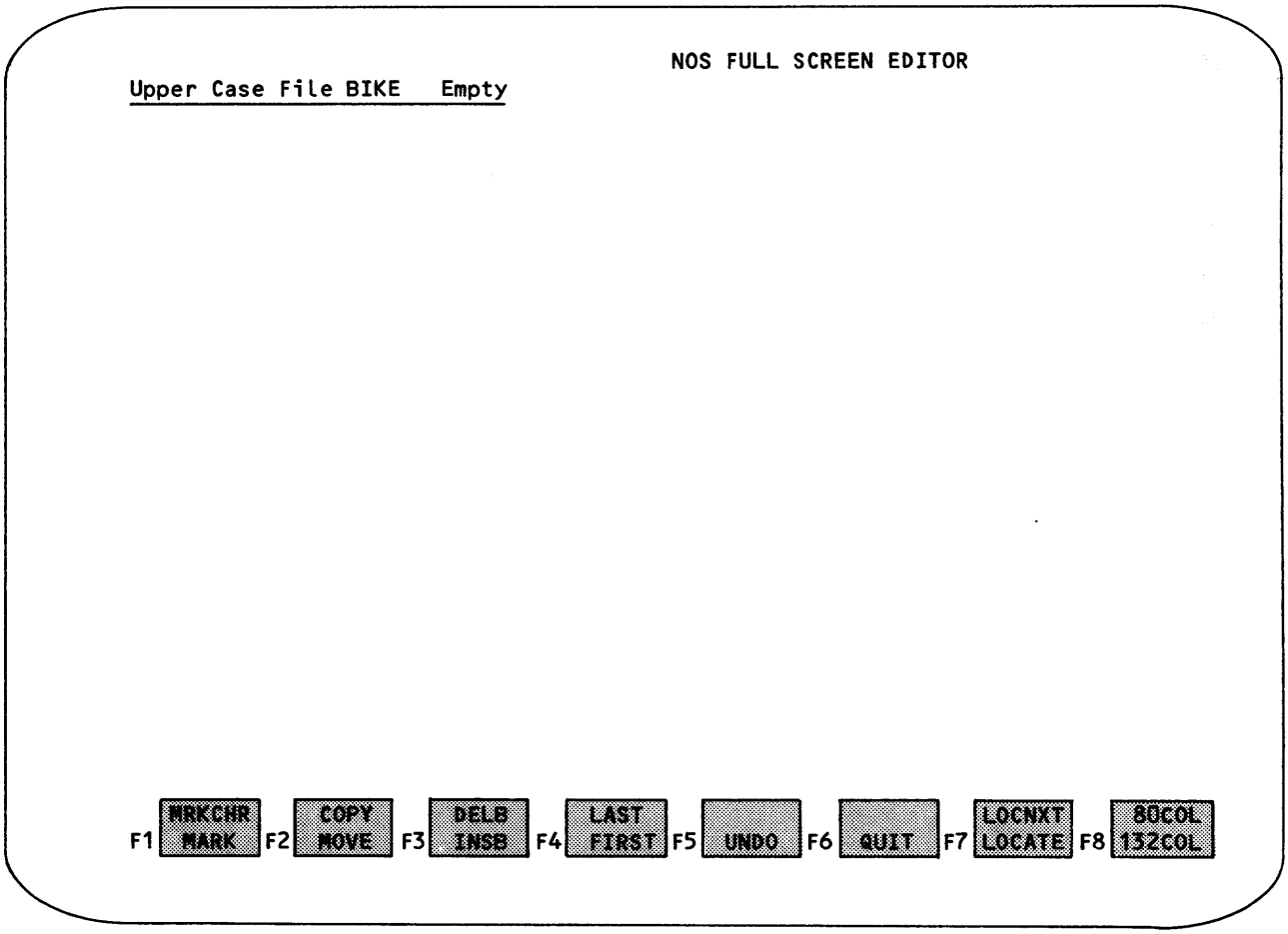


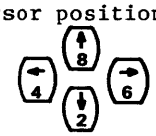
Figure 12-2. FSE Screen

The line:

Upper Case File BIKE Empty

near the top of the screen tells you the name of the file you are editing (the edit file) and that it is empty. Uppercase means that the data you enter in the file will be stored as uppercase characters. The highlighted boxes at the bottom of the screen show you the functions assigned to the programmable function keys. These keys, labeled F1, F2,...,F8, appear at the top of your keyboard. (F1, F2,...,F8 appear below the keys in raised letters.) These keys are described later in this section.

To enter text, position the cursor at the point on the screen where you want the text to appear, and then enter the text. To move the cursor, use the cursor-positioning keys. The cursor positioning keys are on the gray keypad to the right of the main keyboard. They are:



You can change existing text by positioning the cursor on the text to be changed and typing the new text.

ENTERING DIRECTIVES

Most of your basic editing needs can be met by just pressing a few keys. Some functions, however, must be performed by entering FSE directives. FSE directives are instructions that tell FSE what to do.

To enter directives, press:

(HOME)

which is located in the middle of the four cursor-positioning keys. Pressing this key moves the cursor to the FSE directive line at the top of the screen, where you then enter the directive and press:

(NEXT)

Pressing:

(HOME)

does not alter the current cursor position in the edit file. Execution of a directive begins at the position the cursor was in when you pressed:

(HOME)

Most directives and their parameters can be abbreviated to the first letter. In this section, we spell out directives in full to avoid confusion.

For example, to create lists of data in file BIKE you can set the tabs that automatically move the cursor to specified columns on the screen. To set tabs, use the SET directive. Press:

(HOME)

and enter the following, using the numeric keys at the top of the keyboard.

```
SET TAB 6 15 35 55
```

This directive sets tabs at columns 6, 15, 35, and 55.

When entering your data, press:

to move the cursor to the next tab setting. To create the columns shown in figure 11-3, enter each line as follows:

103 fender-front 7/15/83 30

Try entering the information in figure 12-3 in file BIKE.

PART NO	DESCRIPTION	DATE	INVENTORY
103	FENDER-FRONT	7/15/83	30
104	PEDALS-WHITE	7/15/83	30
105	PEDALS-BLACK	7/15/83	14
312	PAINT-BLUE	8/11/83	7
210	FRAME-26IN	7/24/83	18

Figure 12-3. File BIKE

When you are finished working on your file, you can leave FSE by pressing:

and entering:

```
QUIT REPLACE.
```

This ends your FSE session and makes file BIKE an indirect access permanent file. The following appears on your screen.

```
FILE: BIKE (PERMANENT)
FILE: FSEPROC  (NO CHANGES)
```

FSEPROC is a file used by FSE for more advanced editing techniques. We will not be working with it, so you need not worry about it.

EDITING FILES

To edit file BIKE enter the following command.

```
FSE,BIKE,GET.
```

The GET parameter retrieves a copy of indirect access permanent file BIKE and makes it local to your terminal.

The screen in figure 12-4 appears.

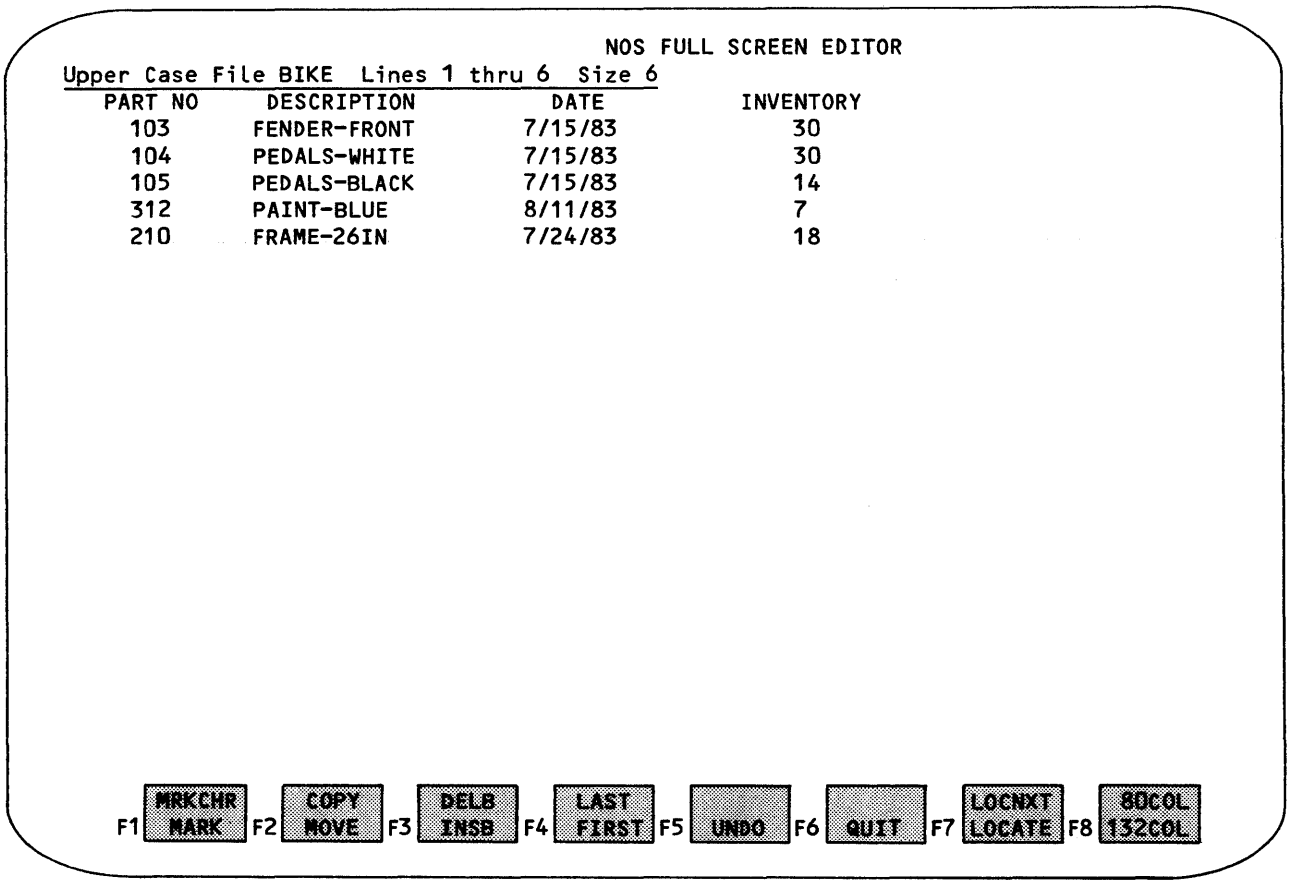


Figure 12-4. File to Be Edited

The line near the top of the screen tells you the name of the file, the lines of the file currently being displayed, and the total number of lines in the file. If your file is uppercase, the line is prefixed by this information.

You can then position the cursor anywhere on the screen to enter changes. You can also use the programmable function keys or directives to perform your editing. Following are brief explanations of how to use some of the keys and directives. We use only a subset of the available parameters for most directives. Refer to the keyboard in figure 11-1 for the location of the keys.

Practice using the following keys and directives by making changes to file BIKE.

Inserting and Deleting

You can insert a space or a line using:

(INSRT)

Pressing the key inserts a space at the current character. If you press the shift key with:

(INSRT)

a blank line is inserted at the current line. You can then enter the new line.

To delete a character or a line, use:

(DLETE)

Pressing the key deletes the current character or space. If you press the shift key with:

(DLETE)

the current line is deleted.

Locating Text

To locate a string of text in your file press:

```
F7 LOCATE
```

The cursor moves to the directive line and FSE prompts you to enter the text you want to locate.

For example, to locate the word BLACK in file BIKE, press:

```
F7 LOCATE
```

The cursor goes to the directive line, and prompts you with:

```
ENTER TEXT
```

To locate the next occurrence of BLACK, enter:

```
BLACK
```

and press:

(NEXT)

This locates the next occurrence from the current cursor position of BLACK in the file. When BLACK is found, the cursor is positioned at the B in BLACK.

Replacing Text

To replace one text string with another, use the REPLACE directive. FSE searches for the specified string and, if found, replaces it with the second specified string. FSE always searches from the current cursor position. The format for the REPLACE directive is:

```
REPLACE/string1/string2
```

string1 is replaced with string2.

For example, to replace the next occurrence of 26IN with 24IN in the following line:

```
210      FRAME-26IN          7/24/83              18
```

press:

(HOME)

and enter:

```
REPLACE /26IN/24IN.
```

and press:

(NEXT)

The line reads:

```
210      FRAME-24IN          7/24/83              18
```

Recovering from Mistakes

You can recover from any mistake you make during your current terminal session. To do this you press

```
F5 UNDO
```

which enables you to undo the effects of the previous editing operation. Successive F5 UNDO entries undo changes you have made to the file in reverse chronological order.

For example, to undo the REPLACE directive described earlier, press:

```
F5 UNDO
```

The change is then undone.

Copying Lines

You can copy one or more lines of your file to another part of the file using:

F1 MARK

and

F1 COPY

(refer to figure 12-1). For example, if you want to copy the following two lines of BIKE:

```
105      PEDALS-BLACK        7/15/83            14
312      PAINT-BLUE          8/11/83            7
```

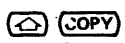
to the end of the file, position the cursor at the first of the two lines. Press:

F1 MARK

which marks and highlights that line as the first line in the group of lines to be copied. Then position the cursor at the second of the two lines. Press:

F1 MARK

again, marking and highlighting that line as the last of a group of lines to be copied. Move the cursor to the last line in the file and press:



or

F2 COPY

Your file then appears as in figure 12-5.

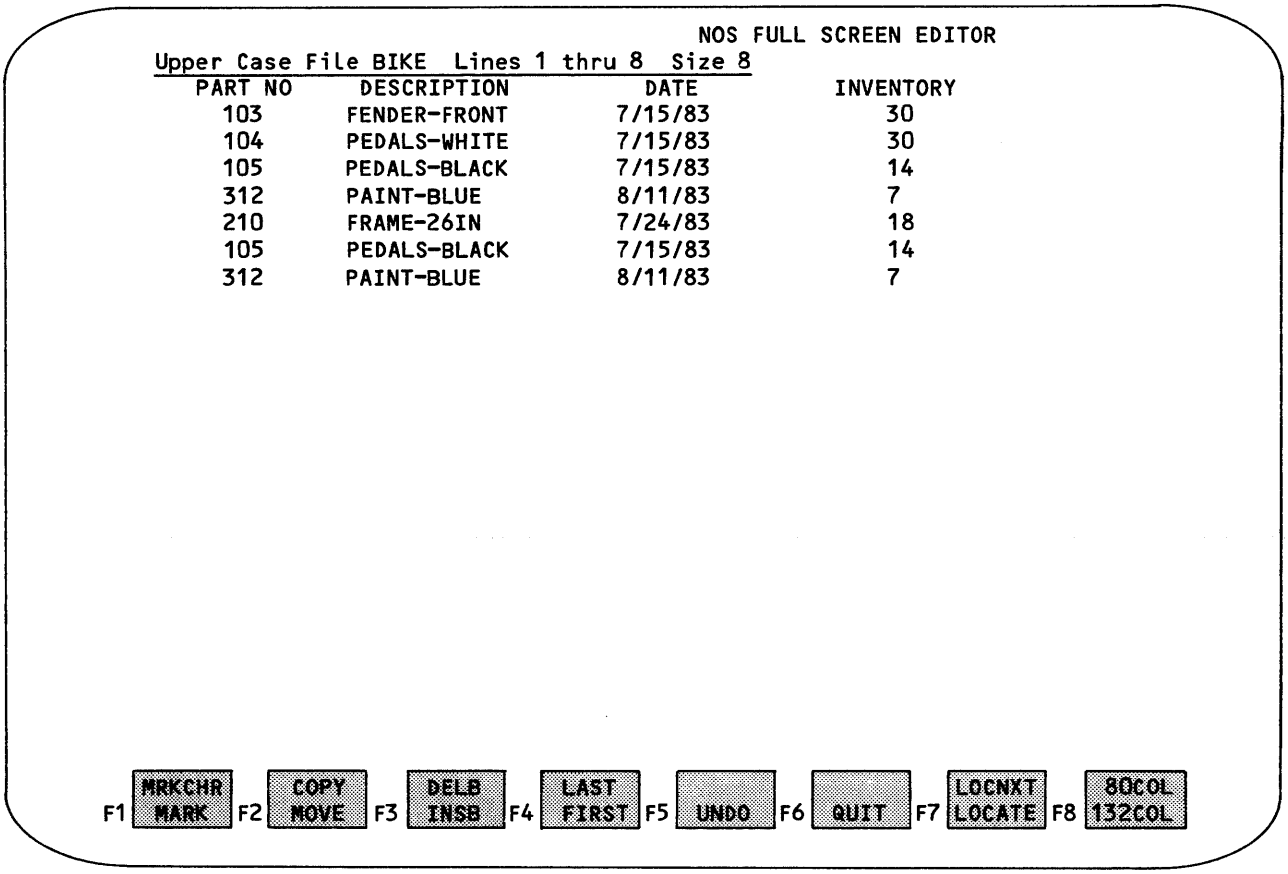


Figure 12-5. Copying Lines

Getting Online Help

If you need help with any of the directives, either press:

(HELP)

or enter the HELP directive. This accesses the HELP file at the beginning of the file. If you need help with one of the directives, enter the HELP directive. The directive format is:

```
HELP directive
```

directive is the name of the directive with which you need help. A description of the specified directive and its parameters is then displayed on the lower half of the screen. When you enter:

```
HELP
```

with no directive parameter, FSE accesses the first screen of the HELP file. You can then independently manipulate the edit file and the HELP file. You can edit your file while the HELP file is being displayed. You can also move forward or backward in either file by positioning the cursor in that section of the screen, and pressing either:

(FWD)

or

(BKW)

(refer to figure 12-1).

For example, to get information on the LOCATE directive, enter:

```
HELP LOCATE
```

The screen in figure 12-6 will appear.

```
Upper Case File BIKE  Lines 1 thru 6  Size 6
  PART NO      DESCRIPTION          DATE          INVENTORY
    103        FENDER-FRONT         7/15/83          30
    104        PEDALS-BLACK         7/15/83          21
    105        PEDALS-WHITE         7/15/83          14
    312        PAINT-BLUE           8/11/83           7
    210        FRAME-24IN           7/24/83          18
    105        PEDALS-BLACK         7/15/83          14
    312        PAINT-BLUE           8/11/83           7


File FSEHELP   Lines 261 thru 272  Size 765
(LOCATE)   (L)  (LN)  (LP)

    Purpose:          Locates a specified string.

    Format:           LOCATE WORD direction/string/range IN tab UPPER QUIET

    Parameters:       WORD
                        Specifies that the string is not part of another word;
                        that is, the string is bounded by punctuation marks,
                        blank characters, beginning of lines, or end of lines.

                      direction
F1 MRKCHR MARK  F2 COPY MOVE  F3 DELB INSB  F4 LAST FIRST  F5 UNDO  F6 QUIT  F7 LOCNXT LOCATE  F8 80COL 132COL
```

Figure 12-6. HELP File for Directive LOCATE

To return from the HELP file to editing only file BIKE, press BACK.

USING PROCEDURES IN SCREEN MODE

You can execute any procedure in either line or screen mode. Screen mode, however, allows you to make full use of the terminal screen to display the parameter prompts and to have the help descriptions for those parameters displayed at the same time. If you have a terminal supported by NOS screen mode, you can execute interactive and menu procedures in screen mode. In this subsection we describe how to use an interactive procedure and a menu procedure in screen mode.

To get started, enter the SCREEN command.

```
/SCREEN,721.
```

The SCREEN command puts your terminal in screen mode. Using FSE, create the procedures in the following subsections and save the procedures on files by the same name. Then you can use these files to practice using procedures in screen mode.

USING AN INTERACTIVE PROCEDURE

Create the following procedure in figure 12-7. You can use this procedure to try the examples in this subsection.

```
.PROC,PROCI*I"COPY AND ROUTE A FILE",
FILEA'File to be copied'=(*F),
FILEB'File copied to'=(*F),
NUMBER'Number of records copied',
WHERE'Routing option'=(IN,LP,PU).
.HELP.

                    This procedure gets two indirect access
                    permanent files and copies a certain
                    number of records from one to the other.
                    Then the file copied onto is queued for
                    input, printing, or punching.
.HELP,FILEA.

                    This parameter is the name of an indirect
                    access permanent file, which will be made
                    local to the job and a certain number of
                    records will be copied from it.
.HELP,FILEB.

                    This parameter is the name of an indirect
                    access permanent file, which will be made
                    local to the job and a certain number of
                    records will be copied onto it.
.HELP,NUMBER.

                    This parameter is the number of
                    records that will be copied
                    from one file to another.
.HELP,WHERE.

                    This parameter is the routing
                    option.  It will queue the file
                    copied onto for input (IN),
                    printing (LP), or punching (PU).
.ENDHELP.
GET,FILEA,FILEB.
COPYCR,FILEA,FILEB,NUMBER.
ROUTE,FILEB,DC=WHERE.
REVERT,NOLIST. COPYBR COMPLETE
```

Figure 12-7. Example of an Interactive Procedure

Save procedure PROCI on a file named PROCI.

Notice that we are using apostrophes instead of quotation marks to enclose the parameter descriptions. By using apostrophes, the parameter keywords, such as FILEA and FILEB, will not appear on the screen when the procedure is called. We will discuss the appearance of the screen more in the subsection Formatting the Screen with Interactive Procedures.

When you call this procedure by entering:

```
PROCI.
```

the screen in figure 12-8 will appear.

```
                    COPY AND ROUTE A FILE

          File to be copied:_____
             File copied to:_____
      Number records copied:_____________________________________
             Routing option:__

                 Specify values and press NEXT when ready

                                                  F5 HELP  F6 QUIT
```

Figure 12-8. Screen Display of Procedure Prompts

The cursor is positioned where you should enter the first parameter. The labeled function keys at the bottom, F5 HELP and F6 QUIT, are used to get help information (discussed later) and to terminate (QUIT) the procedure call. Anytime you want to terminate the procedure call, press the F6 key.

The right side of the screen consists of lines for entering the parameters. Notice that the length of each line is determined by the checklist values in your procedure, such as (*F). For example, the first parameter must be a file name, so the line is seven characters long.

After entering the name of the file you want to copy from, press the tab key to go to the next parameter. Enter each remaining parameter, pressing the tab key after each entry to advance to the next parameter. You can also use the cursor positioning keys to move between parameter fields.

After you have entered all parameters, press:

(NEXT)

and the procedure executes.

If you try to execute the procedure and a required parameter value is missing, the screen reappears with the cursor positioned at the missing parameter and a prompting message appears in the upper right corner. This message is:

```
Please enter
```

followed by the parameter description.

For example, enter file names for the first two parameters (try DATA1 and DATA2 for this example). Then press:

(NEXT)

The screen in figure 12-9 will appear.

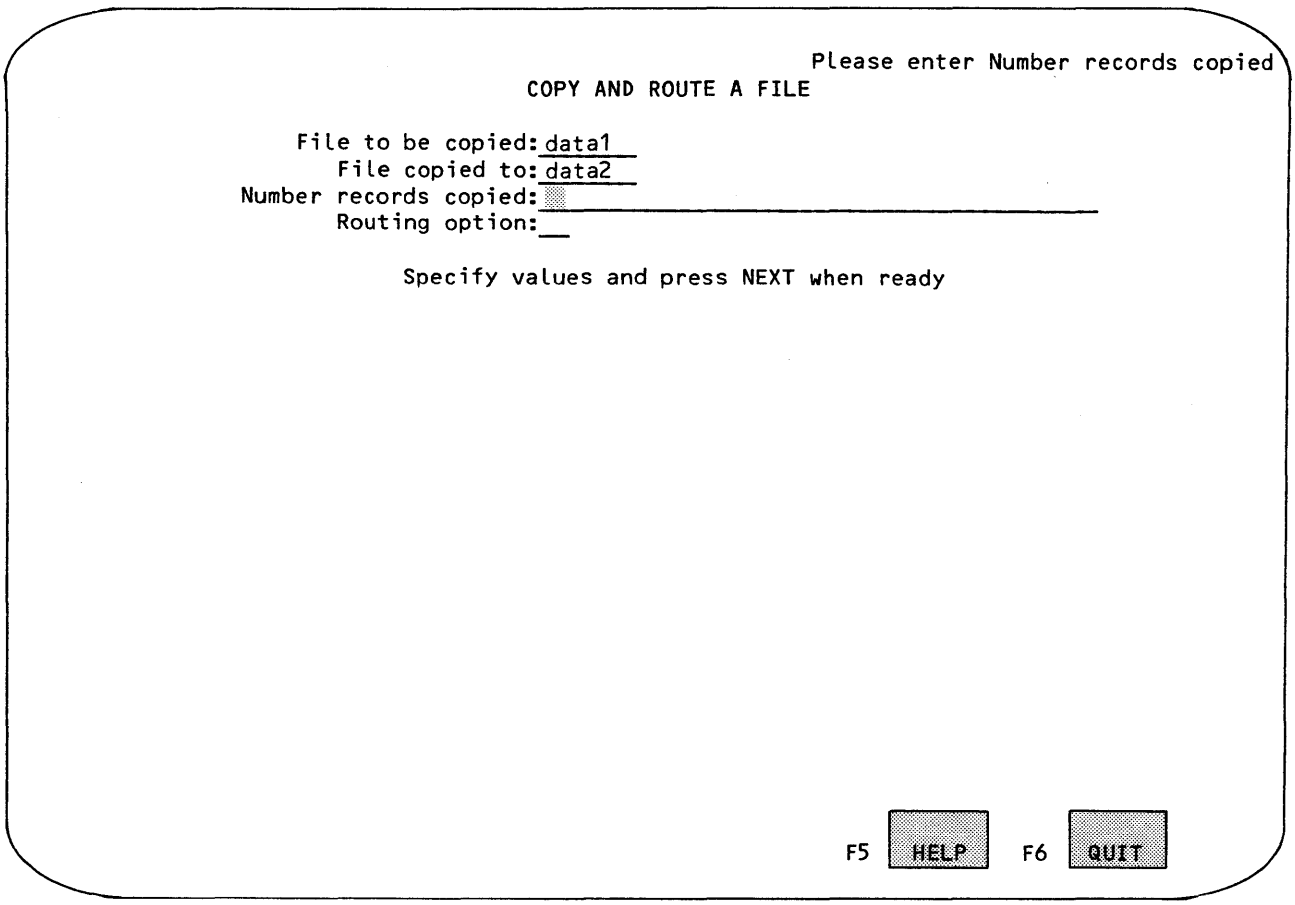


Figure 12-9. Screen Display of Entering Parameters

Enter the remaining two parameters and press:

(NEXT)

to execute the procedure.

If any of the parameters you entered are incorrect, the screen reappears with the cursor positioned at the incorrect parameter. The message in the upper right of the screen prompts you to correct that parameter. Correct the parameter value and press:

NEXT

to execute the procedure.

For example, enter the first three parameters correctly, then enter KR for the routing option. Since KR is not one of the checklist values you wrote in the procedure (IN, LP, PU), it is an incorrect entry. Press:

NEXT

to execute the program and the screen in figure 12-10 appears.

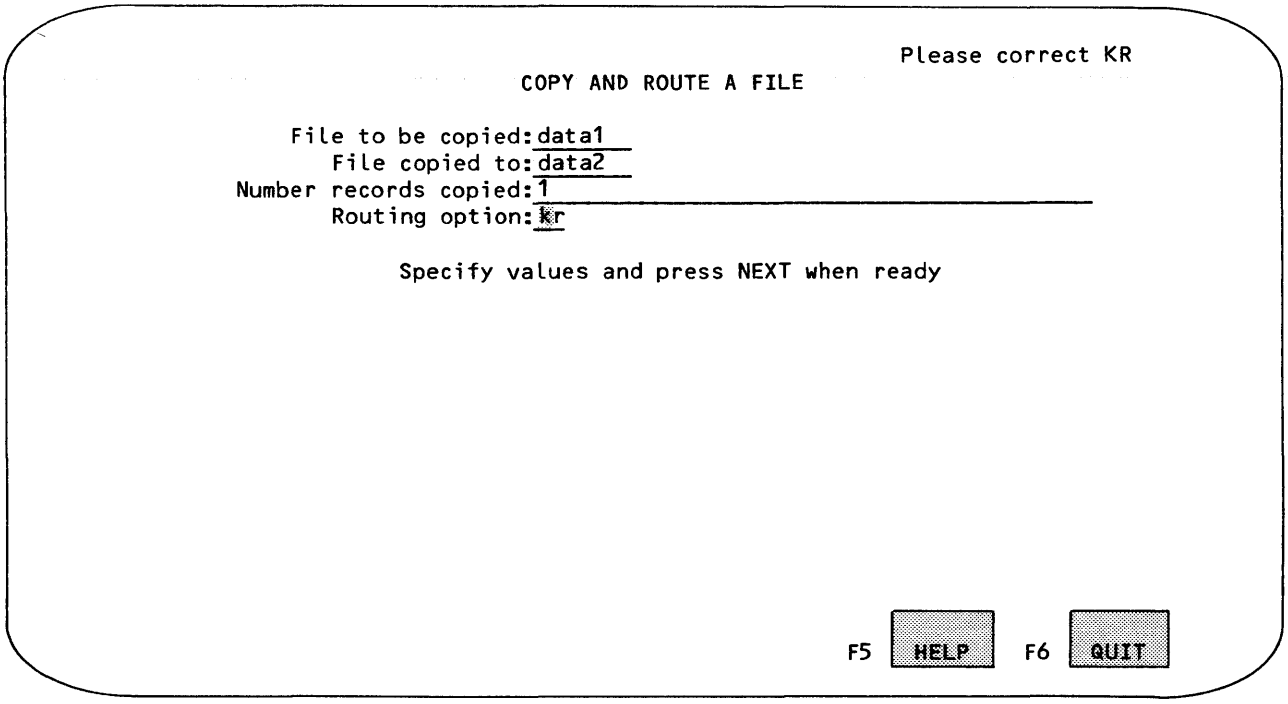


Figure 12-10. Screen Showing an Incorrect Parameter Entry

The cursor is positioned at the routing parameter. Reenter a correct routing option and press:

NEXT

to execute the procedure.

GETTING HELP IN AN INTERACTIVE PROCEDURE

When you are entering the parameters, you can get the HELP descriptions for a parameter or the procedure. When the cursor is on the first parameter line (as in figure 12-11), press:

F5 HELP

(the HELP key can also be used). The screen displays the HELP information for that parameter (figure 12-11).

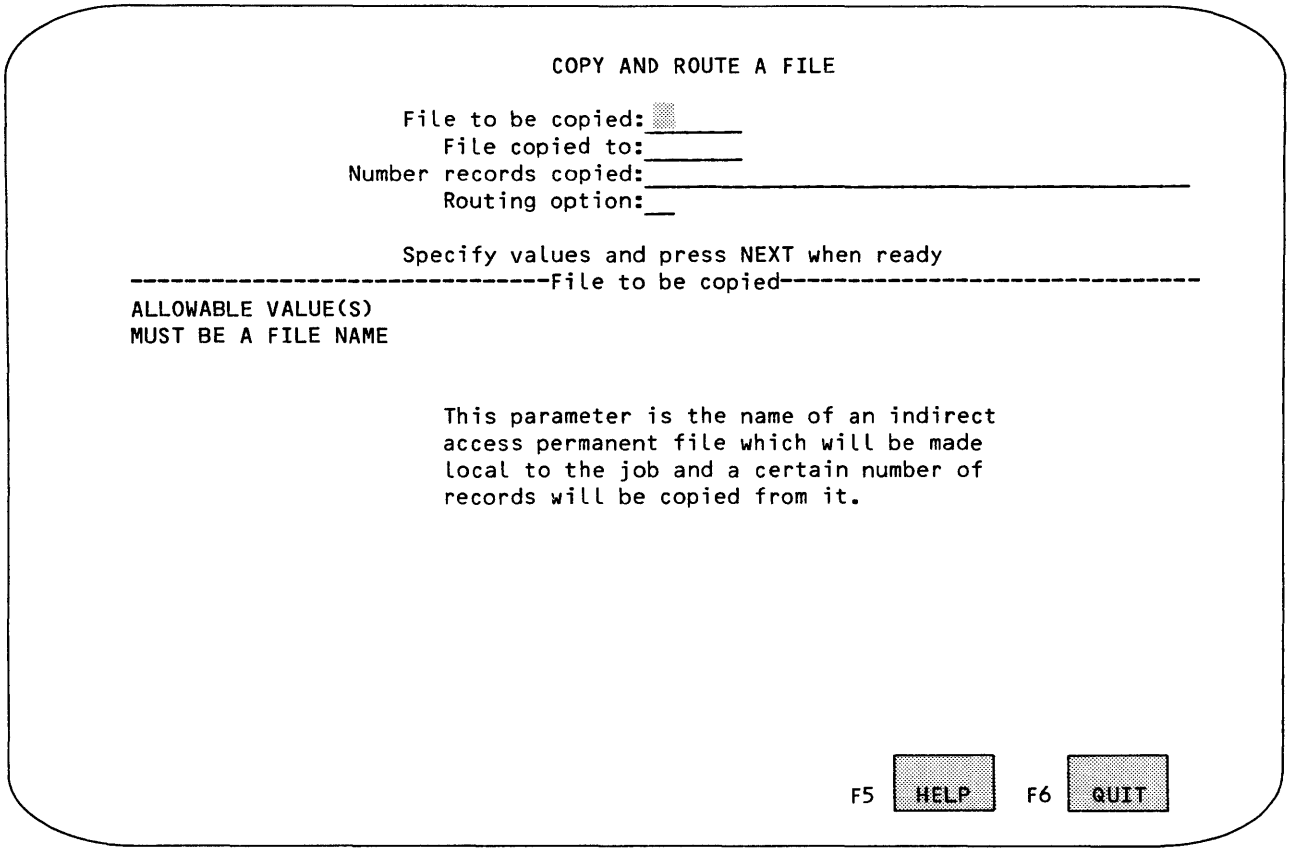

COPY AND ROUTE A FILE
File to be copied:
File copied to:
Number records copied:
Routing option:
Specify values and press NEXT when ready
File to be copied
ALLOWABLE VALUE(S)
MUST BE A FILE NAME
This parameter is the name of an indirect access permanent file which will be made local to the job and a certain number of records will be copied from it.
F5 HELP
F6 QUIT

Figure 12-11. Screen Displaying Parameter HELP Text

The cursor is repositioned at the parameter for which you are getting help, so you can enter the correct information.

Now press the F5 key again, and the help text for the procedure appears (figure 12-12).

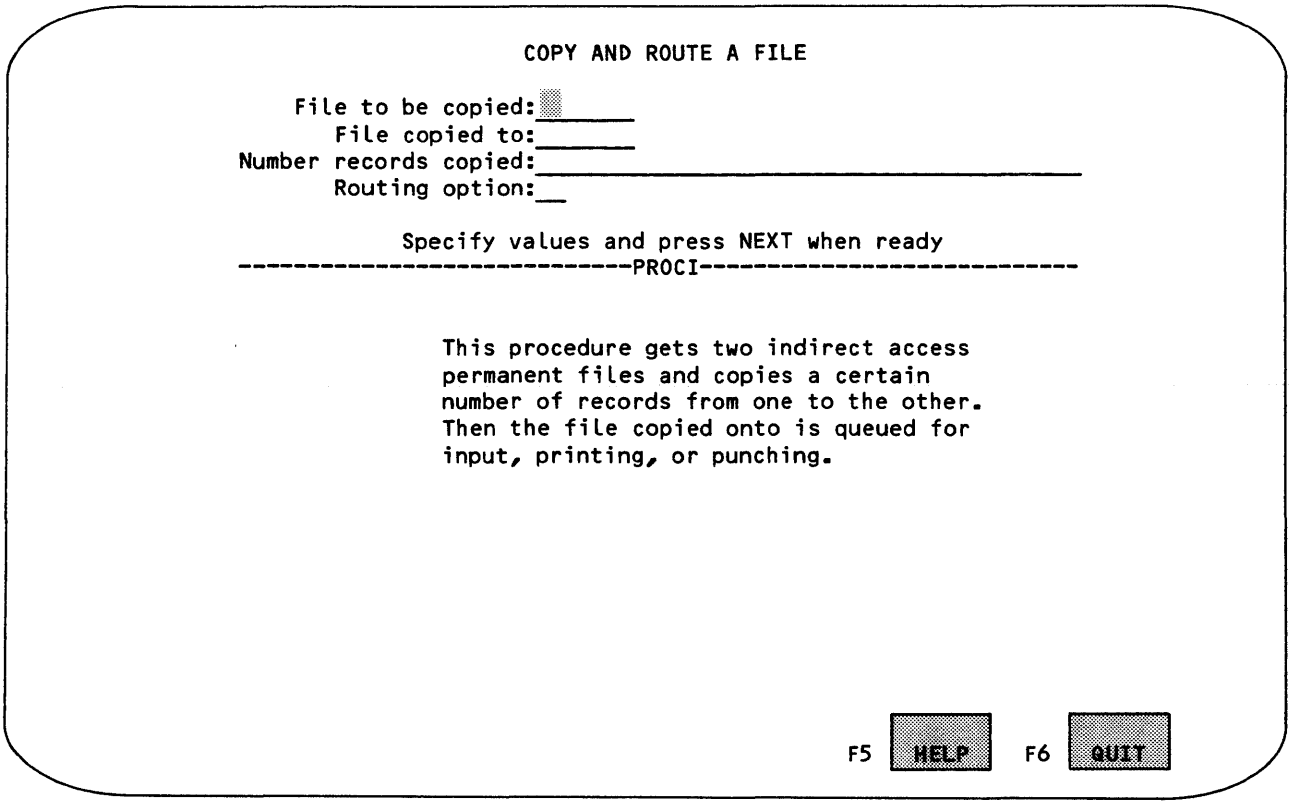

COPY AND ROUTE A FILE
File to be copied:
File copied to:
Number records copied:
Routing option:
Specify values and press NEXT when ready
PROCI
This procedure gets two indirect access permanent files and copies a certain number of records from one to the other. Then the file copied onto is queued for input, printing, or punching.
F5 HELP
F6 QUIT

Figure 12-12. Screen Displaying HELP Text for the Procedure Description

If you continue to press the F5 key, the HELP text for that parameter and the HELP text for the procedure alternately appears.

Move the cursor to any parameter you want, using the tab key or the cursor-positioning keys, and get the HELP description by pressing the F5 key. For example, position the cursor at the routing option, press the F5 key, and the screen in figure 12-13 appears.

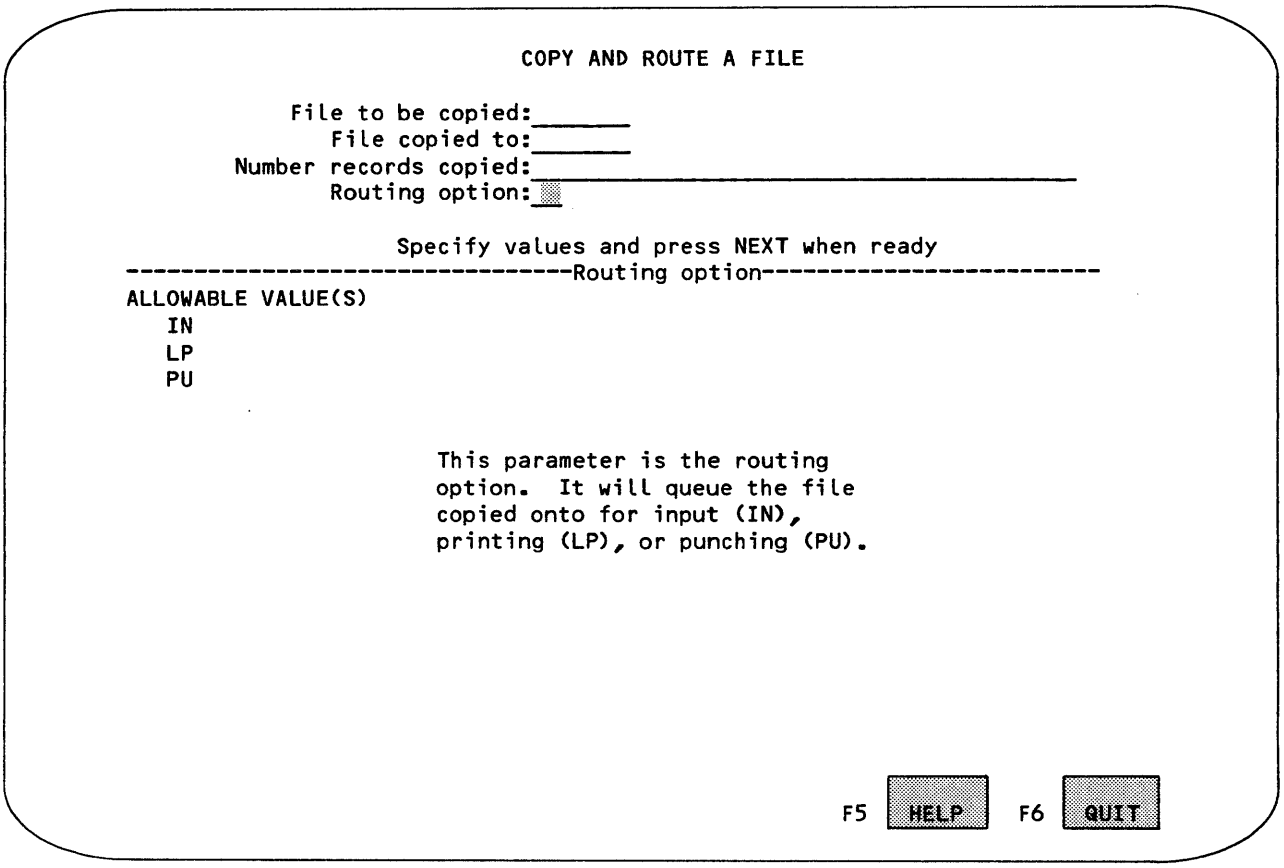

COPY AND ROUTE A FILE
File to be copied:
File copied to:
Number records copied:
Routing option:
Specify values and press NEXT when ready
Routing option
ALLOWABLE VALUE(S)
IN
LP
PU
This parameter is the routing option. It will queue the file copied onto for input (IN), printing (LP), or punching (PU).
F5 HELP
F6 QUIT

Figure 12-13. Screen Displaying Parameter HELP Text

When you no longer need the help information, press:

BACK

and the help text disappears.

FORMATTING THE SCREEN WITH INTERACTIVE PROCEDURES

There are many ways you can control the format of your procedure display on the screen. The descriptions of the keywords will always be displayed on the left half of the screen, and the fields for entering values are on the right. Based on the length of the longest line, the display is centered on the screen. You may want to vary the length of parameter descriptions to improve the screen appearance.

The maximum length for a parameter value is 40 characters. By using the apostrophes instead of quotation marks in the keyword descriptions in the procedure file, the keyword does not appear and, therefore, there is space for more description. If the quotation marks are used instead, the keyword appears before the description and is included in the character count.

If the lengths of the fields for entering values on the right are shorter than 40 characters, you will have space for longer descriptions on the left.

The description determines what appears in the message line in the upper right corner. Forty characters are allowed for the message line, including the characters used for the system-provided prompt:

```
Please enter
```

If your description is too long to fit in the remaining space, it is truncated and an ellipsis is added to the end of the message.

You can also change the text of the system-provided prompts. You can do this in your procedure with special directives. These directives are decribed in Volume 3, System Commands, in the section on procedures.

The way the HELP text appears on the screen is also determined by how you created your procedure. For example, if you created the HELP portion of the procedure with the spaces and margins shown in the example (figure 12-7), your HELP text appears with that spacing on the screen.

By default, the parameter descriptions appear on the screen right-justified with respect to the colons. You can left-justify your descriptions of the parameters if you want. You can do this by adding enough spaces to the end of each description when creating your procedure so that all the description lengths are the same.

USING A MENU PROCEDURE

Create the procedure in figure 12-14. You can use this procedure to try the examples in this subsection.

```
.PROC,MENU*M"FILE ROUTING OPTIONS",OPTION=
(1"Print a file.",
2"Punch a file.",
3"Plot a file.").
.HELP.

                                        This menu procedure gives the user
                                            three different choices in
                                                routing a file.
.HELP,1.

                                              Option 1 routes a file
                                                to a line printer at
                                                 the central site.
.HELP,2.

                                              Option 2 routes a file
                                                to be punched.
.HELP,3.

                                              Option 3 routes a file
                                                to be plotted.
.ENDHELP.

.IF,OPTION.EQ.1.MYPRINT.
.IF,OPTION.EQ.2.MYPUNCH.
.IF,OPTION.EQ.3.MYPLOT.
REVERT,NOLIST.
```

Figure 12-14. Example of a Menu Procedure

Save procedure MENU on a file named MENU.

This procedure allows you to select one of three options for routing a file. It executes procedure MYPRINT, MYPUNCH, or MYPLOT, depending on which selection you make. You can use the procedures that you created with these names in section 8, Creating Menu Procedures.

When you call this procedure by entering:

MENU.

the following screen appears.

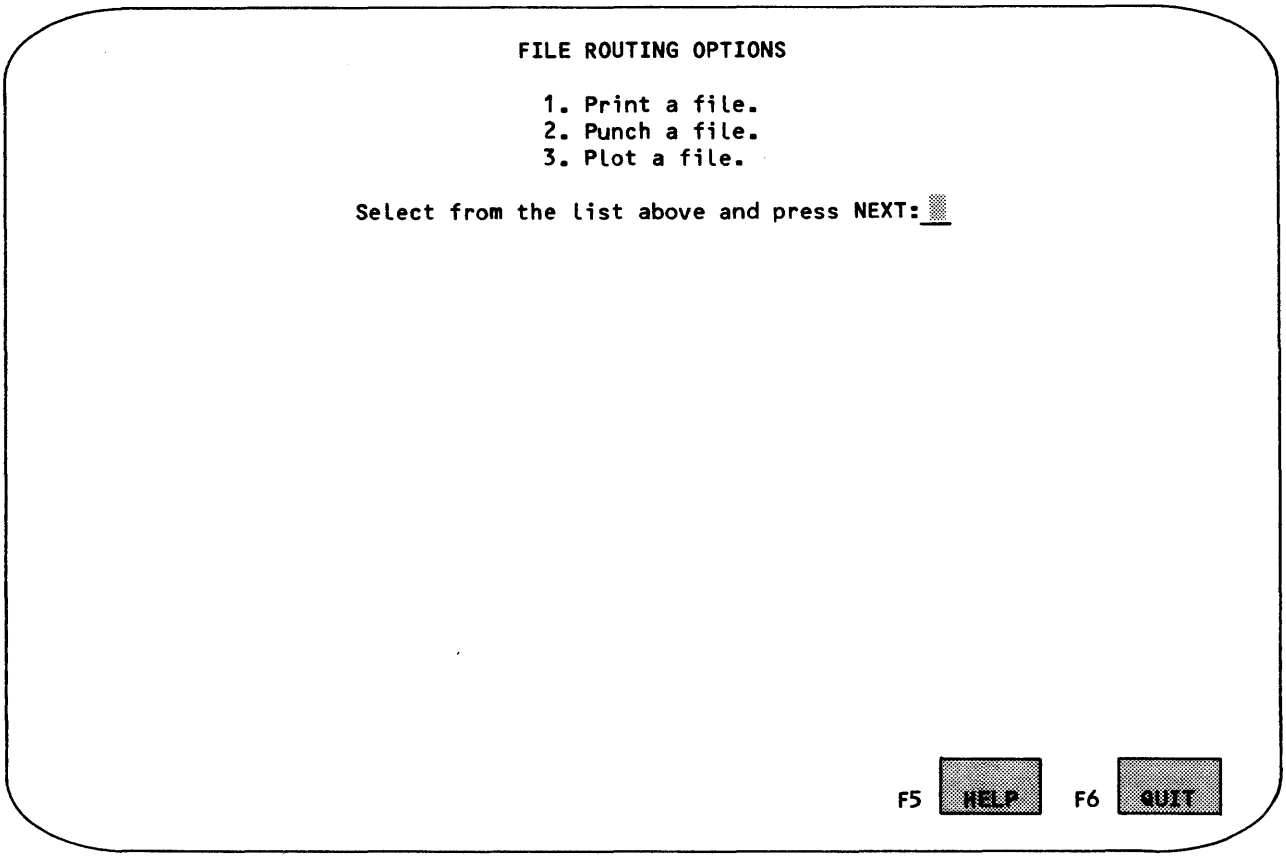


Figure 12-15. Screen Displaying Menu Options

Based on the length of the longest line, the list of selections is centered on the screen.

The cursor is positioned at the input line after the prompt. You can enter your selection (in this case, either 1, 2, or 3) and press:

(NEXT)

to execute that procedure.

If you make an incorrect entry, the screen reappears with the cursor positioned at the input line so you can enter a correct value.

As with interactive procedures, F6 QUIT can be used at any time to terminate the procedure call.

GETTING HELP IN A MENU PROCEDURE

You can get HELP text about the procedure or any of the selections. To get HELP text for the procedure enter a ? on the input line where the cursor is positioned and press NEXT. The screen in figure 12-16 will appear:

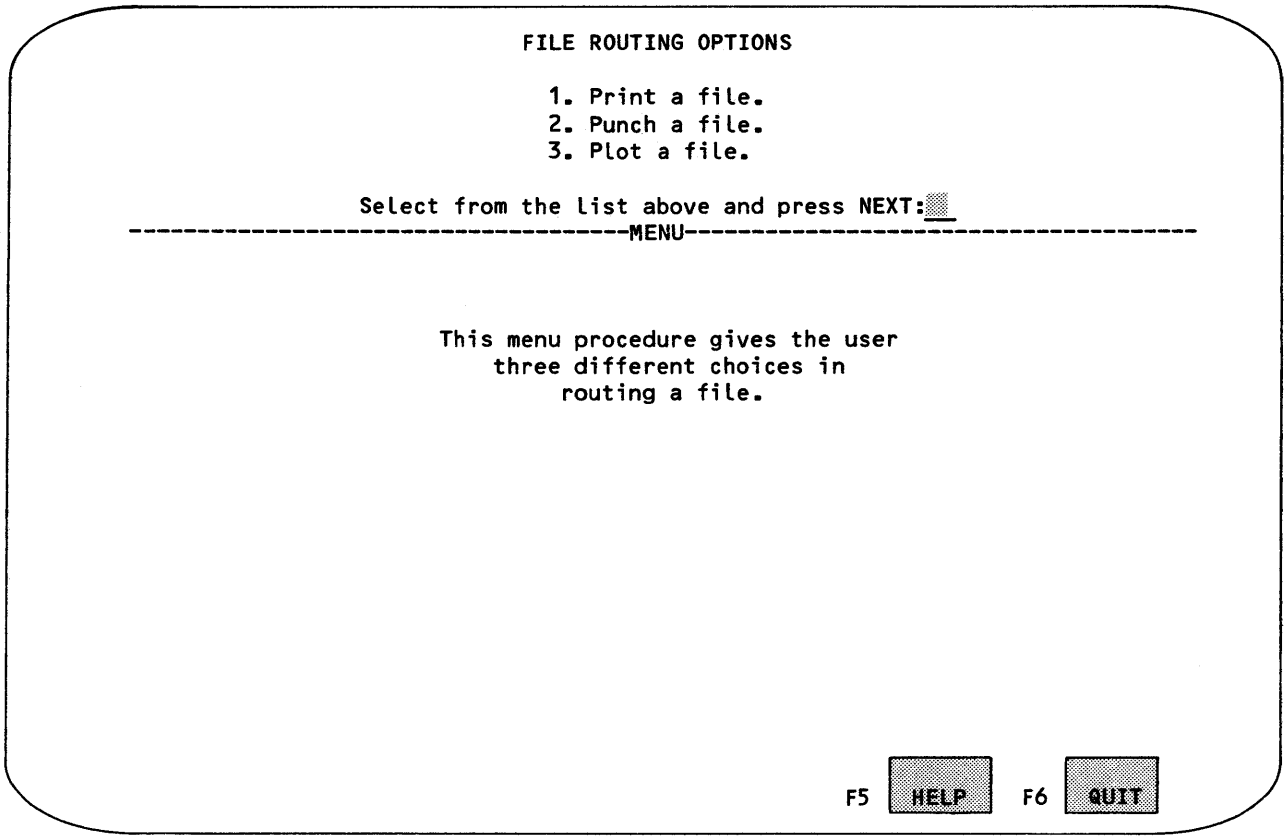

FILE ROUTING OPTIONS
1. Print a file.
2. Punch a file.
3. Plot a file.
Select from the list above and press NEXT:
MENU
This menu procedure gives the user three different choices in routing a file.
F5 HELP
F6 QUIT

Figure 12-16. Screen Displaying Menu HELP Text

You can get the same HELP text by pressing either:

F5 HELP

or:

(HELP)

When you are through with the HELP text, press:

(BACK)

and the HELP text disappears.

To get HELP text for any of the selections, enter the number of that selection on the input line followed by ?. For example, if you enter:

```
1?
```

and press:

NEXT

The screen in figure 12-17 appears.

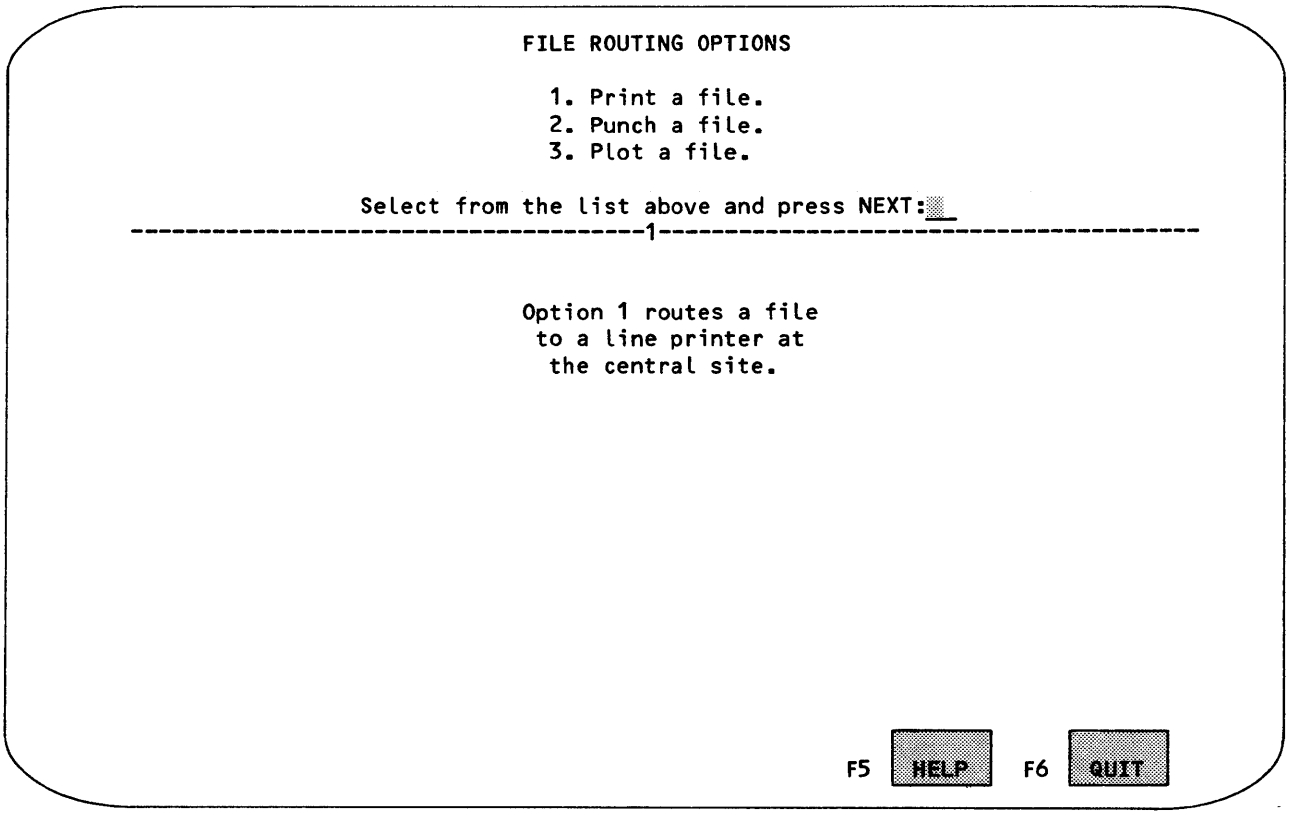

FILE ROUTING OPTIONS
1. Print a file.
2. Punch a file.
3. Plot a file.
Select from the list above and press NEXT:1
Option 1 routes a file
to a line printer at
the central site.
F5 HELP
F6 QUIT

Figure 12-17. Screen Displaying HELP Text for a Menu Option

The HELP text will always appear formatted on the screen as you created it in your procedure.

MULTIPLE-PAGED SCREENS

If you have a procedure with many parameters or selections, the display may fill more than one screen. If this is the case, a page number appears on the screen to the right and below the title.

Also, a third function key:

F1 FWD

appears at the bottom left corner of the screen. You can use this key to page forward in your list of parameters or selections to other pages. As you page forward, another function key:

F2 BKW

appears to the right of F1. You can use this key to page backward in your list.

If the HELP text for a parameter or the procedure is sufficiently long, the system uses more than one page to display it. When there is more than one page of HELP text, the function key:

F3 HELP FWD

appears at the bottom of the screen. You can use this key to page forward in the HELP text. As you page forward, another function key:

F4 HELP BKW

appears at the bottom of the screen, so you can also page backward in the HELP text.

All function keys appear only when there are multiple pages of both the parameter prompts and HELP text. When this is the case, all function key labels appear at the bottom of the screen (figure 12-18).

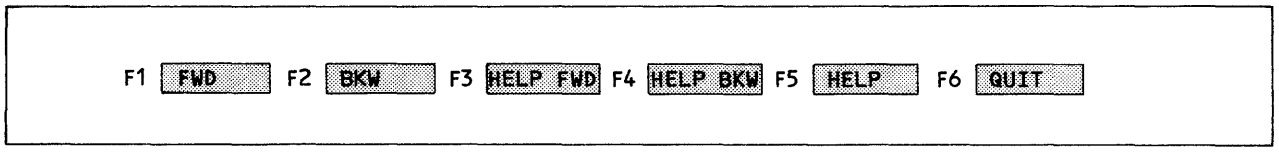


Figure 12-18. Display of Function Keys for Procedures

SOURCE FILE MAINTENANCE 13

The two major methods of changing a file are: using a text editor, such as FSE, or using a source code maintenance utility, such as Modify or CDC Update. Each method has advantages and disadvantages that you must consider in deciding how to maintain your file.

FSE

You can use FSE to change files that contain source programs, data, or text (that is, listable files). With FSE, you can easily maintain a moderately sized file and then compile and execute it with little expense. For a large file, the computer time involved in making changes, and the fact that you must recompile the entire file each time you want to use it, makes using FSE less attractive. However, FSE has some unique capabilities you may want to use: it can search a file for a specific set of characters, change individual characters within a line, and reformat lines.

FSE can be used for either line or screen editing. When line editing, you must edit the file on a line-by-line basis. Line editing is the default mode. When screen editing you can edit a full screen of information. NOS screen mode can support most character mode asynchronous terminal models. Consult your site administration for information on screen mode support for your particular terminal.

The sample terminal session in figure 13-1 shows a few of FSE's capabilities in line mode. The example does not fully explain each command, but gives a general picture of an FSE session. For more information, refer to the NOS Full Screen Editor User's Guide.

Terminal Session	Description
/fse,bike CREATE: BIKE NOS FULL SCREEN EDITOR	The user enters FSE and creates file BIKE.
?? set tab 6 15 35 55	The user sets tabs for columns 6, 15, 35, and 55.
?? insert date inventory 1 ? part no description 2 ? \103\fender-front\7/15/83\30 3 ? \104\pedals-black\7/15/83\21 4 ? \105\pedals-white\7/15/83\14 5 ? \312\paint-blue\8/11/83\7 6 ? \210\frame-26in\7/24/83\18 7 ? carriage return	The user enters the data in file BIKE. The user enters a carriage return to terminate the creating of file BIKE.
?? print 1 quiet	The user returns to the beginning of the file. The quiet directive is used to suppress printing of the line.
?? replace/26/24 6 210 FRAME-24IN 7/24/83 18	The user replaces 26 with 24.
?? print 1 quiet ?? print all 1 PART NO DESCRIPTION DATE INVENTORY 2 103 FENDER-FRONT 7/15/83 30 3 104 PEDALS-BLACK 7/15/83 21 4 105 PEDALS-WHITE 7/15/83 14 5 312 PAINT-BLUE 8/11/83 7 6 210 FRAME-24IN 7/24/83 18	The user again returns to the beginning of the file and then prints all data.
?? copy 4 5 to 6 7 105 PEDALS-WHITE 7/15/83 14 8 312 PAINT-BLUE 8/11/83 7	The user copies lines 4 through 5 after line 6.
Continued on next page	

Figure 13-1. Using FSE (Sheet 1 of 2)

Terminal Session (Continued)					Description
?? delete 8					The user deletes line 8.
8	312	PAINT-BLUE	8/11/83	7	
?? print 1 quiet					The user again returns to the beginning of the file and then prints all data.
?? print all					
1	PART NO	DESCRIPTION	DATE	INVENTORY	
2	103	FENDER-FRONT	7/15/83	30	
3	104	PEDALS-BLACK	7/15/83	21	
4	105	PEDALS-WHITE	7/15/83	14	
5	312	PAINT-BLUE	8/11/83	7	
6	210	FRAME-24IN	7/24/83	18	
7	105	PEDALS-WHITE	7/15/83	14	
?? undo					The user undoes the last executed directive.
?? print 1 quiet					The user again returns to the beginning of the file and then prints all data.
?? print all					
1	PART NO	DESCRIPTION	DATE	INVENTORY	
2	103	FENDER-FRONT	7/15/83	30	
3	104	PEDALS-BLACK	7/15/83	21	
4	105	PEDALS-WHITE	7/15/83	14	
5	312	PAINT-BLUE	8/11/83	7	
6	210	FRAME-24IN	7/24/83	18	
7	105	PEDALS-WHITE	7/15/83	14	
8	312	PAINT-BLUE	8/11/83	7	
?? print 1 quiet					The user returns to the beginning of the file.
?? locate/24					The user locates the string 24.
6	210	FRAME-24IN	7/24/83	18	
?? quit replace					The user stops the FSE session and saves file BIKE.
FILE: BIKE (PERMANENT)					
FILE: FSEPROC (NO CHANGES)					
/					

Figure 13-1. Using FSE (Sheet 2 of 2)

SOURCE CODE MAINTENANCE UTILITIES

The source code maintenance utilities store a file in a compressed format: three or more consecutive blanks are replaced by special code. Compressed files take up less storage space. When using large files, this is an important factor to consider.

Another advantage is that you can specify a group of often used lines to be common. For example, assume that you use FORTRAN statements:

```
COMMON/BLOCK/Y,ARR(50)
INTEGER Y, ARR
```

in more than one subroutine. With Modify or Update, you can put these common statements in one place, rather than with each subroutine. Maintenance is easier because you need only make changes to one set of statements, rather than changing each place they normally occur.

The utilities also keep a history of all the changes, making it easier for you to eliminate previous changes. For example, if you added modifications to a program and they increased the execution time, you could go back to the original program by deleting the statements that modified it.

The main difference between Modify and Update is where they are used: although it may not be this way at your site, most products, such as FORTRAN and COBOL, are maintained in Update format, while the operating system is maintained in Modify format. LIBEDIT is usually used in conjunction with Modify to create and maintain user libraries (LIBEDIT is discussed in section 11).

We don't discuss Update in this guide; for information, refer to the Update Reference Manual.

Introductory information about Modify follows. For complete information, refer to the Modify Reference Manual.

MODIFY CONCEPTS

Before you can use Modify, you must understand:

- What prepared source files and program libraries are.
- How to use directives.
- What output files are available to you.
- What the general structure of a Modify job is.

Explanations of these concepts follow.

PREPARED SOURCE FILES

Modify requires that you prepare your source file so that it can identify where you want to make changes. You do this by grouping lines within the file and giving each group, called a deck, a name (the deck name). You may also identify lines that are common to more than one part of the file and designate them as a common deck.

PROGRAM LIBRARIES

Modify uses your prepared source file to create a compressed file, called the program library. This is the file you use when you make changes.

Besides deleting blanks, Modify associates a name and number with each line in the program library. The names are the deck names you specify in the prepared source file. The numbers show the order of the lines within a deck (1 is the first line in the deck). For example, in the deck named PAD, the following association occurs:

Lines in Deck	Associated Name and Number
APPLE, A	PAD 1
BANANA, A	PAD 2
ORANGE, O	PAD 3
GRAPE, G	PAD 4
PEACH, G	PAD 5

As you make changes to the program library, you specify a modification name for each set of changes you make (described later). Modify associates this name with the changed line and gives it a new number. The numbers show the order of lines within the modification set. For example, after changing lines 2 and 5 of deck PAD under the modification name PAPER the association is:

Lines in Changed Deck	Associated Name and Number
APPLE, A	PAD 1
BANANA, B	PAPER 1
ORANGE, O	PAD 3
GRAPE, G	PAD 4
PEACH, P	PAPER 2

DIRECTIVES

Directives are the special instructions you use to create and modify a program library. When using Modify from batch, you put the directives in a separate record after the commands. When using Modify interactively, you enter directives in response to prompts from the system. For both batch and interactive jobs, you also can put the directives in a separate file and name the file in the MODIFY command (for more information on using a separate file, refer to the Modify Reference Manual). A directive's format is:

```
*dirname p1,p2,...,pn
```

Entry	Description
*	The prefix character; it must be in column 1. You can change the prefix character, but in this guide the asterisk is used.
dirname	The directive name; it must start in column 2. This guide describes a subset of the directives.
p_i	Optional directive parameters. You must separate the directive name and the first parameter with a character other than: A through Z 0 through 9 : + - * / ( ) $ = Commas must separate subsequent parameters. In this guide, either a comma or one or more spaces separate the directive name and subsequent parameters.

In this guide, the formats shown for the directives conform to the NOS coding standard, which specifies that file names, modification names, and deck names on Modify directives start in column 11. This standard provides a consistent framework for making changes and also provides for easy reading.

OUTPUT FILES

Modify produces several files as output, all of which are optional. You specify which files you want generated when you access Modify. In an interactive job, the compile file is the only file automatically produced; in a batch job, you also automatically get responses generated by Modify.

Following are descriptions of some of the output files.

File	Description
Compile	A file that contains the information in your prepared source file (including subroutines called), the changes you made using Modify, and the identifier Modify associated with each line. You can use this file as input to a compiler or assembler and as data for an application program.
New program library	A file that contains the same information as the compile file, but the information is in program library compressed format. You use this file when making additional changes in another Modify session.
Source-text	A file that contains the information in your prepared source file and any changes you made using Modify. This file has the deck names you added, but it does not have the sequence numbers and modification set names that Modify associates with each line. If you want to convert a program library back to source format, you can use the source-text file.

GENERAL MODIFY JOB STRUCTURE

You can use Modify in both batch and interactive jobs. Refer to the examples in figure 13-2 to familiarize yourself with the general structure of each. Individual commands and directives are described later.

Interactive Format	Description
/GET,lfn	The user makes the file that Modify will use a local file.
/MODIFY,parameters ? directive ? directive ? directive ? directive ? carriage return	The user accesses Modify, enters one directive after each prompt, and enters a carriage return to terminate the directives.
	By default, Modify lists any errors that have occurred. Either you can have this and other information put on a separate file that can be printed later or you can decide not to print it at all (refer to the L and LO parameters on the MODIFY command in the Modify Reference Manual). If no errors occur, Modify prints: MODIFICATION COMPLETE.

Figure 13-2. General Structures for Batch and Interactive Jobs (Sheet 1 of 2)

Batch Job Format	Description
ujn. USER,username,password. CHARGE,*.	Job identification.
GET,lfn.	The command that makes the file Modify will use a local file.
MODIFY,parameters.	The command that accesses Modify.
7/8/9 directive . . . directive 6/7/8/9	The record containing Modify directives.
	By default, Modify lists the following information: Some directives Deck status Errors Modifications performed Text input Refer to the L and LO parameters on the MODIFY command in the Modify Reference Manual for further information.

Figure 13-2. General Structures for Batch and Interactive Jobs (Sheet 2 of 2)

USING MODIFY

Before using Modify, you must prepare your file. After preparing the file, you call Modify to create a program library, to generate a compile file and to make changes to the program library. Since your prepared file is only used the first time you call Modify, you can remove it from the system after creating the program library.

PREPARING YOUR SOURCE FILE

The first step in preparing your source file is to divide the file into decks by putting a deck name as the first line of a deck and an EOR as the last line (the deck name and the EOR are called boundaries). Modify can then change lines within a specified group. Although Modify does not limit deck size, each deck should be a manageable number of lines. Within a program, for example, you could make the program one deck and each subroutine an additional deck. The format for deck names is:

Format	Description
deckname	A 1- to 7-character deck name beginning in column 1. The characters can be: A through Z, 0 through 9, or + - * / ( ) $ =

Figure 9-3 is an example of a FORTRAN program before and after adding deck names and EORs. Either you can use FSE to add these boundaries (refer to the NOS Full Screen Editor User's Guide) or you can include them in your card deck.

Original File	Prepared File	Description
	MAIN	The deck name.
PROGRAM EXAMPLE	PROGRAM EXAMPLE	
INTEGER ODD	INTEGER ODD	
PRINT*,'LIST N ODD NUMBERS'	PRINT*,'LIST N ODD NUMBERS'	
1 PRINT*,'HOW MANY NUMBERS?'	1 PRINT*,'HOW MANY NUMBERS?'	
READ(*,*,END=10) ODD	READ(*,*,END=10) ODD	
CALL LIST (ODD)	CALL LIST(ODD)	
GO TO 1	GO TO 1	
10 STOP	10 STOP	
END	END	
	--EOR--	The end of the first deck.
	SUB1	The deck name.
SUBROUTINE LIST (ODD)	SUBROUTINE LIST(ODD)	
INTEGER ODD	INTEGER ODD	
DO 5 I=1,2*ODD,2	DO 5 I=1,2*ODD,2	
PRINT*, I	PRINT*, I	
5 CONTINUE	5 CONTINUE	
RETURN	RETURN	
END	END	

Figure 13-3. Example of Deck Names and EORs

To create a common deck, you place the set of common lines at the beginning of your file and add the word COMMON in columns 1 through 6 of the second line of the deck (as for all decks, the first line is the deck name, and the last line is EOR).

Also, you put a call directive in each place in the file where the common lines apply. The CALL directive's format is:

```
*CALL deckname
```

Parameter	Description
deckname	Name of the deck that contains the common lines.

Modify replaces the CALL directive with the lines in the common deck when it creates the compile file; otherwise, it treats the CALL directive as a line within the file.

An example of a FORTRAN program that has a common deck is shown in figure 13-4. The compile file is the same as the original file.

Original File	Prepared File	Description
	COMEVEN	The common deck named COMEVEN.
	COMMON	
	COMMON/EVEN/EVEN	
	INTEGER EVEN	
	--EOR--	The end of the common deck.
	FIRST	The deck name.
PROGRAM PRINT	PROGRAM PRINT	
COMMON/EVEN/EVEN	*CALL COMEVEN	The directive that calls common deck COMEVEN.
INTEGER EVEN		
PRINT*,'THE LIST'	PRINT*,'THE LIST'	
CALL NUMBER	CALL NUMBER	The statement that calls subroutine NUMBER.
.	.	
.	.	
.	.	
END	END	
	--EOR--	The end of deck FIRST.
	SECOND	The deck name.
SUBROUTINE NUMBER	SUBROUTINE NUMBER	
COMMON/EVEN/EVEN	*CALL COMEVEN	The directive that calls common deck COMEVEN.
INTEGER EVEN		
.	.	
.	.	
.	.	
END	END	

Figure 13-4. Example of a Common Deck

CALLING MODIFY

Once you have prepared your source file, you are ready to begin using Modify. To access Modify, you use the MODIFY command. The command's format is:

MODIFY,p_1,p_2,...,p_n.

All parameters are optional and can be listed in any order. Explanations of the parameters you are most likely to use follow.

CREATING A PROGRAM LIBRARY

Before you can make changes to the file using Modify, you must create a program library. When using Modify to make changes, you will use this program library, not the source file. When you access Modify with this intention, we recommend that you use the following parameters on the MODIFY command.

Parameter	Description
N=filename	Specifies the local file† on which Modify is to create the new program library. You must make this file permanent after the Modify run if you want to make additional modifications in a subsequent job. filename is often called the new program library. If you specify only N, Modify creates the program library on a file named NPL.
P=0	Specifies that you are not using a program library as input to Modify (you are using a prepared source file).
F	Directs Modify to edit all decks. Modify writes all the decks that are in the prepared file on the program library file named in parameter N=filename.

After accessing Modify, you direct it to create a program library with the CREATE directive. Its format is:

*CREATE file

Parameter	Description
file	Name of the local input source file.

†If the file used to create the program library has two or more identical deck names, Modify uses only the last deck with that name, placing it where the first deck with that name was located.

The example in figure 13-5 shows an interactive terminal session in which Modify creates program library NPROG from source file SFILE. For this and all following examples, we assume that the user has logged in under the batch subsystem in IAF.

Terminal Session	Description
/get,sfile	The user makes the source file a local file.
/modify,n=nprog,p=0,f	The user specifies NPROG as the program library.
? *create sfile	The user uses prepared file SFILE.
? carriage return	The user specifies that all directives have been entered.
MODIFICATION COMPLETE.	Modify signals that it has created the program library.
/replace,nprog	The user saves file NPROG for later use.

Figure 13-5. Creating a Program Library

It is not necessary to create the program library in a separate run as shown in figure 9-5. You can enter the CREATE directive and other modification directives during the same Modify session or have them in the same record of a batch job. You can also create all output files in one session or job.

You can also create a program library from other program libraries, specifying which decks you want to use (refer to the Modify Reference Manual for information).

GENERATING A COMPILE FILE

When accessing Modify with the intention of obtaining a compile file containing all decks, we recommend that you use the following parameters on the MODIFY command.

Parameter	Description
C=filename	Compile file output parameter; Modify writes the compile file on file filename. If you omit this parameter or specify only C, Modify writes the compile file on file COMPILE.
P=filename	Program library input parameter; filename is the local file that contains the program library input and is often called the old program library. Modify created this file in a previous run. At that time it was called the new program library and was named by the N=filename parameter. It consists of the old program library from that run, created program libraries, and all modifications produced during the run. Modify uses this file as a basis for generating the compile file, adding to it any modifications you make during this Modify run. If you omit this parameter or specify only P, Modify assumes the program library you want to use is on local file OPL. The old program library must be a local file. Make the file a local file before entering the MODIFY command.
F	Modify compiles the entire file (all decks).

The example in figure 13-6 shows an interactive session that generates a FORTRAN compile file and then executes the file.

Typically, you will use a binary file more than once. Instead of recreating the binary file each time, you can make the file permanent. In figure 13-6 the user created the binary file LGO. To save that file you could enter

```
REPLACE,LGO=BINPROG.
```

BINPROG then becomes the permanent file containing the same information as is on LGO.

Terminal Session	Description
/get,nprog	The user makes file NPROG a local file.
/modify,p=nprog,c,f	The user indicates she or he wants a compile file generated using old program library NPROG.
? directive ? directive . . . ? directive ? carriage return	The user enters modification directives for changes to file NPROG and then enters a carriage return. If no changes are necessary, the user enters only a carriage return.
MODIFICATION COMPLETE.	Modify signals that it made the changes to file NPROG and then generated a compile file named COMPILE.
/ftn5,i=compile,l=0 . . . program information . . .	The user enters the FORTRAN command using file COMPILE as the input. The system returns the program information created by FORTRAN to the terminal.
/lgo . . . program output . . .	The user runs the program.

Figure 13-6. Generating a Compile File

MAKING CHANGES TO THE PROGRAM LIBRARY

You direct Modify to make changes to the program library by entering modification directives. This guide describes some of these directives; for a complete list, refer to the Modify Reference Manual.

Before entering the MODIFY command, you must decide on the mode in which you want to make changes. The two modes we discuss are full edit and selective edit; refer to the Modify Reference Manual for a discussion of other modes.

When you are in full edit mode, Modify writes all decks available to it to the output files you specify on the MODIFY command, which means that you can incorporate changes into any of the decks. You select full edit mode by specifying F on the MODIFY command.

If you use selective edit mode, Modify writes only those decks you specify to the output files listed on the MODIFY command. You choose selective edit mode by omitting the F parameter on the MODIFY command and specifying the EDIT directive during the Modify session. On the EDIT directive you indicate which deck or decks you want written to the output files. Modify writes them to the output files in the form they are in when it encounters the directive. The EDIT directive's format is:

*EDIT $p_1,p_2,\ldots,p_n$

Parameter	Description
p_i	A deck name or range of deck names in one of the following forms: deckname or $deckname_a.deckname_b$ The first form specifies one deck name on the program library; the second form specifies a range of deck names, starting with $deckname_a$ and ending with $deckname_b$. Entering a deck name that is not on the program library or giving a range of deck names in the wrong order causes an error.

Modify places all decks except the decks you create during this Modify session on the output files in the order in which the names appear on the EDIT directives. The new decks are placed at the end of the new program library, but are placed in the order their names appear on the EDIT directives for the compile and source text files. If you are modifying a common deck, place its deck name first on the EDIT directive to ensure that all files receive the latest version of the common deck (refer to the Modify Reference Manual for detailed information on common decks).

When you want to make changes to your program library, the parameters you specify on the MODIFY command are dependent upon whether you have a program library before entering Modify and what types of output files you want generated, as well as what edit mode you want. The following list briefly describes your options.

Parameter	Description
C=filename	Names a compile file for output.
C=0	Specifies not to generate compile file.
N=filename	Names a new program library for output.
P=filename	Names an old program library as input.
P=0	Specifies that a program library is not used as input.

Figure 13-7 shows the use of the MODIFY command and the EDIT directive.

Terminal Session	Description
/get,oldp	
/modify,p=oldp,n=newp	The user wants to modify program library OLDP in selective edit mode and to create a new program library NEWP.
? directive ? directive . . . ? directive	The user enters modification directives that cause Modify to change one or more of the decks.
? *edit chair,table	The user copies decks CHAIR and TABLE to output files. Modifications to decks CHAIR and TABLE that were entered before the EDIT directive are also copied.
? directive ? directive . . . ? directive	The user enters other modification directives to change other decks.
? *edit spoon.fork ? carriage return MODIFICATION COMPLETE.	The user copies decks SPOON through FORK to output files. Modifications made before the EDIT directive are included.
/replace,newp	The user saves file NEWP.

Figure 13-7. Using the EDIT Directive

You identify each set of changes you make by assigning a unique name to the set. Because Modify records the changes under this name, you can take out the entire set of changes at one time. The IDENT directive assigns the name to the modification set. Its format is:

```
*IDENT    modname
```

Parameter	Description
modname	One- to seven-character modification name you want assigned to this modification set. Omitting modname or using a name that has already been used causes an error.

You can use one IDENT directive for several decks or several IDENT directives for one deck. After you enter the IDENT directive, you specify the deck in which changes will be made. The DECK directive identifies the name of this deck. Its format is:

```
*DECK    deckname
```

Parameter	Description
deckname	The name of the deck where you want to make changes. Omitting the deck name or specifying a name that is not one of the deck names in the file causes an error.

After entering the IDENT and DECK directives, you specify the modification directives for the changes you want to make.

The example in figure 13-8 shows the relationship between the IDENT and DECK directives.

Terminal Session		Description
/modify,f		
? *ident	mod1	The user associates the name MOD1 with the changes that follow the DECK directive.
? *deck	loop	The user specifies LOOP as the deck where the changes that follow are to be made.
? directive		The user enters changes for deck LOOP (the changes are identified by MOD1).
? directive		
.		
.		
.		
? directive		
? *deck	results	The user specifies RESULTS as the deck where the changes that follow are to be made.
? directive		The user enters changes for deck RESULTS (the changes are identified by MOD1).
? directive		
.		
.		
.		
? directive		
? *ident	mod2	The user associates the name MOD2 with the changes that follow.
? *deck	loop	The user specifies LOOP as the deck where the changes are to be made (changes can be made to one deck in more than one modification set).
? directive		The user enters changes for deck LOOP (the changes are identified by MOD2).
? directive		
.		
.		
.		
? directive		
? carriage return		
MODIFICATION COMPLETE.		

Figure 13-8. Using the IDENT and DECK Directives

Adding and Deleting Lines

The INSERT and DELETE directives are modification directives that allow you to add and delete lines from the program library.

Before you can enter the INSERT or DELETE directives, you must know the location of the line within the program library where you want to make changes. Modify lists the location identifiers on the compile file (discussed in Generating a Compile File).

When you first create the program library, the location of a line is the deck name you put in the source file and the sequence number, assigned by Modify, that gives the line's position within the deck (refer to figure 13-9).

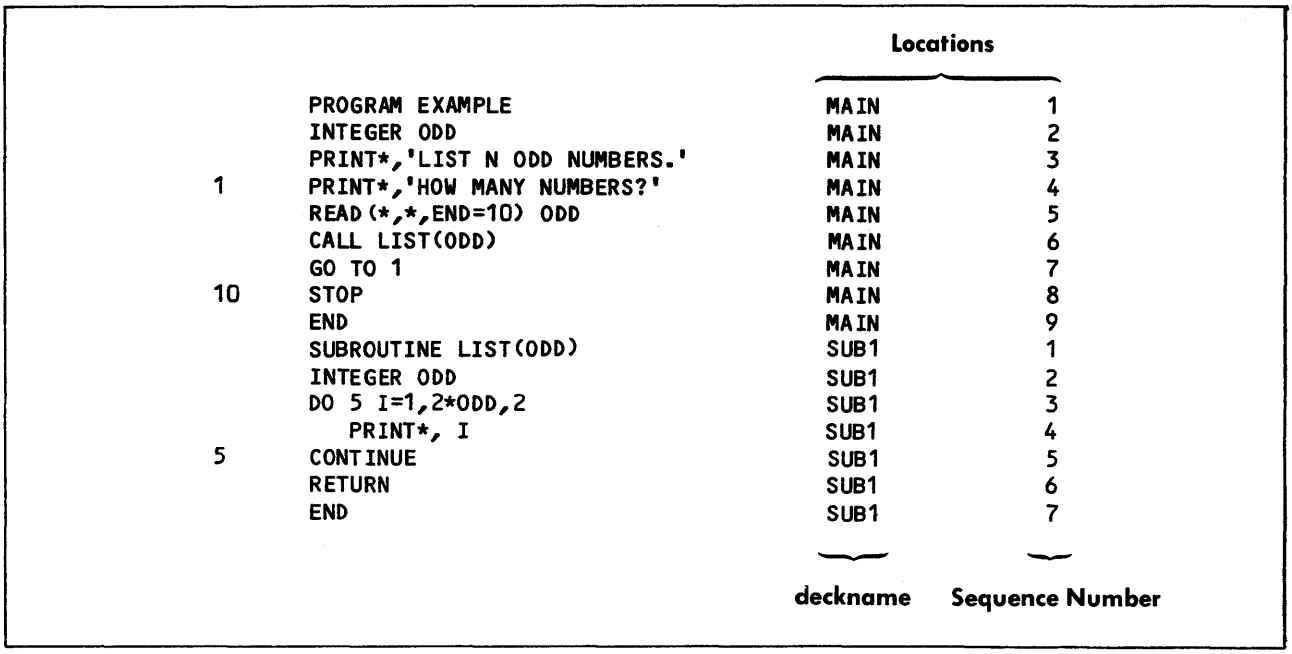


Figure 13-9. Locating Lines within the Compile File

The lines you change or add during modification have different location identifiers: the modification name you give in the IDENT directive and a sequence number related to the order in which the change was made in a given deck. For example, if you enter:

```
*IDENT    CHANGE1
*DECK     MAIN
```

and then:

1. Change **PROGRAM EXAMPLE** to **PROGRAM NUMBER**.

2. Add the line:

   ```
   PRINT*,'PRESS CR to STOP.'
   ```

   after line:

   ```
   4,PRINT*,'HOW MANY NUMBERS?'
   ```

Modify generates the compile file in figure 13-10.

```
                                          Locations
                                    ------------------------
      PROGRAM NUMBER                CHANGE1          1   First change
      INTEGER ODD                   MAIN             2
      PRINT*,'LIST N ODD NUMBERS.'  MAIN             3
1     PRINT*,'HOW MANY NUMBERS?'    MAIN             4
      PRINT*,'PRESS CR TO STOP.'    CHANGE1          2   Second change
      READ(*,*,END=10) ODD          MAIN             5
      CALL LIST(ODD)                MAIN             6
      GO TO 1                       MAIN             7
10    STOP                          MAIN             8
      END                           MAIN             9
      SUBROUTINE LIST(ODD)          SUB1             1
      INTEGER ODD                   SUB1             2
      DO 5 I=1,2*ODD,2              SUB1             3
        PRINT*, I                   SUB1             4
5     CONTINUE                      SUB1             5
      RETURN                        SUB1             6
      END                           SUB1             7
                                    ----------       ---
                                    deckname or modname  Sequence Number
```

Figure 13-10. Locating Lines within a Modified Compile File

To add lines to the program library, you enter the INSERT directive and, immediately after, the lines you want added. The next modification directive ends the insertion. The INSERT directive's format is:

```
*INSERT,c
  or
*I,c
```

Parameter	Description
c	Identifies the location of the line after which the new lines are to be added. If the line is identified by the deck name, c is just the line's sequence number. If the line was added or changed during a modification to the original deck, c must have the following format. modname.number modname is the name of the modification set that added or changed the line (refer to the IDENT directive). number is the sequence number in the modification set. Any attempt to use a line that is not in the deck named in the most recent DECK directive causes an error.

The abbreviated form:

```
*I,c
```

is used in the NOS coding standard. For a complete description of all possible ways of entering the INSERT directive, refer to the Modify Reference Manual. The example in figure 13-11 shows a compile file before a user inserts some lines, a session where lines are inserted, and the resulting compile file (the file has two decks, PARTS and FRAME, and one modification set, HORN).

COMPILE FILE BEFORE MODIFICATION

```
PART NO    DESCRIPTION      DATE        INVENTORY      PARTS     1
  102      FENDER-BACK      8/12/82        20          HORN      1
  103      FENDER-FRONT     7/15/82        30          PARTS     2
  104      PEDALS-BLACK     7/15/82        21          PARTS     3
  105      PEDALS-WHITE     7/15/82        14          PARTS     4
  210      FRAME-27IN       8/12/82        18          HORN      1
  215      HANDLE BARS      7/24/82        23          FRAME     2
```

MODIFY SESSION

Terminal Session				Description
/modify,p=bikepl1,c=bikec2,f				
? *ident bell				The user specifies the modification set name BELL.
? *deck parts				The user specifies deck PARTS for modification.
? *i,4				The user adds the lines after the line with sequence number 4 in deck PARTS. The new lines are identified by BELL 1 and BELL 2.
? 110	chain-27in	8/20/82	17	
? 120	seat-banana	8/20/82	8	
? *deck frame				The user specifies deck FRAME for modification.
? *i,horn.1				The user adds the line after the line with sequence number 1 of modification set HORN in deck FRAME. The new line is identified by BELL 1 (since the modification is in a different deck, the numbering starts at 1 again).
? 212	frame-24in	8/20/82	10	
? carriage return				
MODIFICATION COMPLETE.				

COMPILE FILE AFTER MODIFICATION

```
PART NO    DESCRIPTION      DATE        INVENTORY      PARTS     1
  102      FENDER-BACK      8/12/82        20          HORN      1
  103      FENDER-FRONT     7/15/82        30          PARTS     2
  104      PEDALS-BLACK     7/15/82        21          PARTS     3
  105      PEDALS-WHITE     7/15/82        14          PARTS     4
  110      CHAIN-27IN       8/20/82        17          BELL      1
  120      SEAT-BANANA      8/20/82         8          BELL      2
  210      FRAME-27IN       8/12/82        18          HORN      1
  212      FRAME-24IN       8/20/82        10          BELL      1
  215      HANDLE BARS      7/24/82        23          FRAME     2
```

Figure 13-11. Using the INSERT Directive

To delete lines from the program library, you use the DELETE directive. Modify, however, does not physically delete the lines from the program library but only logically deactivates them. To replace lines with one or more new lines, you add the new lines immediately after the DELETE directive. The next modification directive ends those lines being added. The DELETE directive's formats are:

```
*DELETE,c
*DELETE,c1,c2
 or
*D,c
*D,c1,c2
```

Parameter	Description
c	Identifies the location of the line to be deactivated. If the line is identified by the deck name in the DECK directive, c is just the line's sequence number. If the line was added or changed during a modification to the original deck, c must have the following format. modname.number modname is the name of the modification set that changed or added the line (refer to the IDENT directive). number is the sequence number in the modification set. Any attempt to deactivate a line that is not in the deck named in the most recent DECK directive causes an error.
c_1,c_2	Identifies the first and last lines of a group of lines to be deactivated within a deck. c_1 must occur before c_2 and both c_1 and c_2 must have the same modname. Any lines within the group that are already inactive are not affected. c_1 and c_2 have the same formats as c.

The abbreviated formats:

```
*D,c
*D,c1,c2
```

are used in the NOS coding standard.

For a complete description of all possible ways of entering the DELETE directive, refer to the Modify Reference Manual. The example in figure 13-12 shows a compile file before a user deactivates some lines, a session where lines are deactivated, and the resulting compile file (the file has three decks, MAIN, FIRST, and SECOND, and one modification set, COW).

COMPILE FILE BEFORE MODIFICATION

```
      PROGRAM DAIRY(OUTPUT)                          MAIN          1
    C MAIN PART OF PROGRAM                           MAIN          2
      PRINT*,'    BEGIN DAIRY'                       MAIN          3
    C FIRST SUBROUTINE CALL                          MAIN          4
      CALL MILK                                      MAIN          5
    C SECOND SUBROUTINE CALL                         MAIN          6
      CALL CHEESE                                    COW           1
      PRINT*,'    END DAIRY'                         MAIN          8
      STOP                                           MAIN          9
      END                                            MAIN         10
      SUBROUTINE MILK                                FIRST         1
      PRINT*,'COLLECT MILK,'                         COW           1
      PRINT*,'STORE MILK,'                           FIRST         3
      RETURN                                         FIRST         4
      END                                            FIRST         5
      SUBROUTINE CHEESE                              SECOND        1
      PRINT*,'PROCESS MILK,'                         SECOND        2
      PRINT*,'AGE CHEESE.'                           SECOND        3
      RETURN                                         SECOND        4
      END                                            SECOND        5
```

MODIFY SESSION

Terminal Session	Description
/modify,p=dairyp2,c=dairyc2,f	
? *ident colby	The user gives the modification set the name COLBY and specifies deck MAIN for modification.
? *deck main	
? *d,6	The user deletes and replaces line 6 in deck MAIN.
? c cheese deleted	
? *d,cow.1	The user deletes line 1 of modification set COW.
? *deck second	The user specifies deck SECOND for modification.
? *d,1,5	The user deletes lines 1, 2, 3, 4, and 5 in deck SECOND.
? carriage return	
MODIFICATION COMPLETED.	

COMPILE FILE AFTER MODIFICATION

```
      PROGRAM DAIRY(OUTPUT)                          MAIN          1
    C MAIN PART OF PROGRAM                           MAIN          2
      PRINT*,'    BEGIN DAIRY'                       MAIN          3
    C FIRST SUBROUTINE CALL                          MAIN          4
      CALL MILK                                      MAIN          5
    C CHEESE DELETED                                 COLBY         1
      PRINT*,'    END DAIRY'                         MAIN          8
      STOP                                           MAIN          9
      END                                            MAIN         10
      SUBROUTINE MILK                                FIRST         1
      PRINT*,'COLLECT MILK,'                         COW           1
      PRINT*,'STORE MILK,'                           FIRST         3
      RETURN                                         FIRST         4
      END                                            FIRST         5
```

Figure 13-12. Using the DELETE Directive

Deleting Effects of Modification Sets

The YANK directive nullifies the effect of modification sets you entered previously. It affects the lines of the modification set in the following ways.

- If a line was added by the modification set, Modify deactivates it.
- If a line was deactivated by the modification set, Modify reactivates it.
- If a line was changed by the modification set, Modify restores it to what it was before.

The YANK directive's format is:

```
*YANK      modname
```

Parameter	Description
modname	Name of a modification set previously applied to decks in the program library. Omitting modname or specifying a name that is not a modification set name causes an error.

The YANK directive causes one modification set to be deactivated. To deactivate more than one modification set, you must enter a YANK directive for each modification set.

The example in figure 13-13 shows the use of the YANK directive.

COMPILE FILE BEFORE USING THE YANK DIRECTIVE

```
AAAA                                    INK     1
AAAA1                                   MOD1    1
AAAA                                    INK     3
AAAA4                                   MOD4    1
BBB                                     PEN     1
BBB2                                    MOD2    1
BBB2                                    MOD2    2
CC1                                     MOD1    1
CC                                      BLOT    2
D                                       PAD     1
D3                                      MOD3    1
```

MODIFY SESSION

Terminal Session	Description
/modify,p=inkp11,c=inkc2,f ? *yank mod1 ? carriage return MODIFICATION COMPLETE.	The user deactivates modification set MOD1.

COMPILE FILE AFTER USING THE YANK DIRECTIVE

```
AAAA                                    INK     1
AAAA                                    INK     2
AAAA                                    INK     3
AAAA4                                   MOD4    1
BBB                                     PEN     1
BBB2                                    MOD2    1
BBB2                                    MOD2    2
CC                                      BLOT    1
CC                                      BLOT    2
D                                       PAD     1
D3                                      MOD3    1
```

Figure 13-13. Using the YANK Directive

CHARACTER SETS A

A character set is composed of graphic and control characters. A code set is a set of codes used to represent each character within a character set.

A graphic character may be displayed at a terminal or printed by a line printer. Examples are the characters A through Z and the digits 0 through 9. A control character initiates, modifies, or stops a control operation. An example is the backspace character that moves the terminal carriage or cursor back one space. Although a control character is not a graphic character, a terminal may produce a graphic representation when it receives a control character.

All references within this manual to the ASCII character set or the ASCII code set refer to the character set and code set defined in the American National Standard Code for Information Interchange (ASCII, ANSI Standard X3.4-1977). References in this manual to the ASCII character set do not necessarily refer to the ASCII code set.

NOS supports the following character sets.

- CDC graphic 64- (or 63-) character set.
- ASCII 128-character set.
- ASCII graphic 64- (or 63-) character set.
- ASCII graphic 95-character set.

Each installation selects either the 64-character set or the 63-character set. The differences between the two are described in Character Set Anomalies in this appendix. Any reference in this appendix to the 64-character set implies either the 63- or 64-character set, unless otherwise stated.

NOS supports the following code sets.

- 6-bit display code.
- 6/12-bit display code.
- 7-bit ASCII code.

Display code is a set of 6-bit codes from 00_8 to 77_8.

The 6/12-bit display code is a combination of 6-bit codes and 12-bit codes. The 6-bit codes are 00_8 through 77_8, excluding 74_8 and 76_8. (Refer to Character Set Anomalies for the interpretation of the 00_8 and 63_8 codes.) The 12-bit codes begin with either 74_8 or 76_8 and are followed by a 6-bit code. Thus, 74_8 and 76_8 are considered escape codes and are never used as 6-bit codes within the 6/12-bit display code set. The 12-bit codes are 7401_8, 7402_8, 7404_8, 7407_8, and 7601_8 through 7677_8. All other 12-bit codes ($74xx_8$ and 7600_8) are undefined.

The 7-bit ASCII code (as defined by ANSI Standard X3.4-1977) is right-justified in a 12-bit byte. Assuming that the bits are numbered from the right starting with 0, bits 0 through 6 contain the ASCII code, bits 7 through 10 contain zeros, and bit 11 distinguishes the 0000_8 code from the end-of-line byte. The 7-bit codes are 0001_8 through 0177_8 and 4000_8.

CHARACTER SET ANOMALIES

NOS interprets two codes differently when the installation selects the 63-character set rather than the 64-character set. In tables A-1, A-2, and A-3, the codes for the colon and percent graphic characters in the 64-character set are unshaded; the codes for the colon and percent graphic characters in the 63-character set are shaded.

If an installation uses the 63-character set, the colon graphic character is always represented by a 63_8 code, and the 00_8 code is undefined. However, if the installation uses the 64-character set, output of 6/12-bit display codes 7404_8 or 6-bit display code 00_8 produces a colon. In ASCII mode for interactive jobs, a colon can be input only as a 7404_8 6/12-bit display code.

When using either the 63- or 64-character set, the use of undefined 6/12-bit display codes in output files produces unpredictable results and should be avoided.

On input, NOS recognizes alternate 029 punch codes of 11-0 for the right bracket (]) and 12-0 for the left bracket ([). The alternate codes support the COBOL sign overpunch convention and are not recommended for other uses. Refer to COBOL 5 Reference Manual.

Also, two 00_8 codes may be confused with an end-of-line byte and should be avoided (refer to Volume 3, System Commands for further explanation).

Translation of 7-bit ASCII to 6-bit display code causes character set folding from the 128-character ASCII set to the 63- or 64-character ASCII subset. The following special character substitutions occur:

7-Bit ASCII		6-Bit Display Code		7-Bit ASCII	
Code	Character	Code	Character	Code	Character
0140	`	74	@	0100	@
0173	{	61	[	0133	[
0174	| ⟶	75	\ ⟶	0134	\
0175	}	62	]	0135	]
0176	~	76	^	0136	^

CHARACTER SET TABLES

This appendix contains character set tables for interactive jobs, batch jobs, and jobs involving magnetic tapes. Table A-1 is for interactive jobs, and table A-2 is for batch jobs. Table A-3 is a conversion table used to cross-reference 7-bit ASCII codes and 6/12-bit display codes and to convert ASCII codes from octal to hexadecimal.

Tables A-4, A-5, and A-6 list the magnetic tape codes and their display code equivalents.

The character set tables are designed so that you can find the character represented by a code (such as in a dump) or find the code that represents a character. To find the character represented by a code, look up the code in the column listing the appropriate code set and then find the character on that line in the column listing the appropriate character set. To find the code that represents a character, you first look up the character and then find the code on the same line in the appropriate column.

INTERACTIVE JOBS

Table A-1 shows the character sets and code sets available to you at an ASCII code terminal. When in normal mode (specified by the NORMAL command), NOS displays the ASCII graphic 64-character set and interprets all input and output as 6-bit display code. When in ASCII mode (specified by the ASCII command), NOS displays the ASCII 128-character set and interprets all input and output as 6/12-bit display code.

To determine the octal or hexadecimal ASCII code for a character, refer to table A-3. (Certain terminal definition commands require specification of an ASCII code.)

On output, the US code is reserved for network use and defined as an end-of-line. Use of this character, except in transparent mode, causes incorrect formatting and possible loss of output characters.

BATCH JOBS

Table A-2 lists the CDC graphic 64-character set, the ASCII graphic 64-character set, and the ASCII graphic 95-character sets. It also lists the code sets and card punch codes (026 and 029) that represent the characters.

The 64-character sets use display code as their code set; the 95-character set uses 7-bit ASCII code. The 95-character set is composed of all the characters in the ASCII 128-character set that can be printed at a line printer (refer to jobs using Line Printers). Only 7-bit ASCII code files can be printed using the ASCII graphic 95-character set. To print a 6/12-bit display code file (usually created by an interactive job in ASCII mode), you must convert the file to 7-bit ASCII code. To do this, you enter the FCOPY command. The 95-character set is represented by 7-bit ASCII codes 0040_8 through 0176_8.

JOBS USING LINE PRINTERS

The batch character set printed depends on the print train used on the line printer to which the file is sent (refer to Volume 3, System Commands). The following are the print trains corresponding to each of the batch character sets.

Character Set	Print Train	PSU Printer Print Band
CDC graphic 64-character set	596-1	-
ASCII graphic 64-character set	596-5	530-1
ASCII graphic 95-character set	596-6	530-2

The characters of the default 596-1 print train are listed in the table A-2 column labeled CDC Graphic (64 Char); the 596-5 print train characters are listed in the table A-2 column labeled ASCII Graphic (64 Char); and the 596-6 print train characters are listed in the table A-2 column labeled ASCII Graphic (95 Char).

If a transmission error occurs when printing a line, the system prints the line again. The CDC graphic print train prints a concatenation symbol ( ) in the first printable column of the repeated listing of the line. The ASCII print trains print an underline (_) instead of the concatenation symbol.

If an unprintable character exists in a line (that is, a 7-bit ASCII code outside the range 0040_8 through 0176_8), the number sign (#) appears in the first printable column of a print line, and a space replaces the unprintable character.

To route and correctly print a 6/12-bit display code file on a line printer with the ASCII graphic 95-character set, you must convert the 6/12-bit display code file to a 7-bit ASCII code file with the FCOPY command (refer to Volume 3, System Commands). The resulting 7-bit ASCII file can be routed to a line printer (refer to Volume 3, System Commands) but cannot be output at an interactive terminal.

Table A-1. Character Sets for Interactive Jobs (Sheet 1 of 2)

ASCII Graphic (64 Character)	ASCII Character (128 Character)	6-Bit Display Code	6/12-Bit Display Code	7-Bit ASCII Code	ASCII Graphic (64 Character)	ASCII Character (128 Character)	6-Bit Display Code	6/12-Bit Display Code	7-Bit ASCII Code
: colon		00†			# number sign	# number sign	60	60	0043
Display code 00 is undefined at sites using the 63-character set.					[opening bracket	[opening bracket	61	61	0133
A	A	01	01	0101	] closing bracket	] closing bracket	62	62	0135
B	B	02	02	0102	% percent sign	% percent sign	63	63†	0045
C	C	03	03	0103	: colon	: colon	63	63	0072
D	D	04	04	0104	" quote	" quote	64	64	0042
E	E	05	05	0105	_ underline	_ underline	65	65	0137
F	F	06	06	0106	! exclamation point	! exclamation point	66	66	0041
G	G	07	07	0107	& ampersand	& ampersand	67	67	0046
					' apostrophe	' apostrophe	70	70	0047
H	H	10	10	0110	? question mark	? question mark	71	71	0077
I	I	11	11	0111	< less than	< less than	72	72	0074
J	J	12	12	0112	> greater than	> greater than	73	73	0076
K	K	13	13	0113	@ commercial at	@ commercial at	74†	7401	0100
L	L	14	14	0114	\ reverse slant	\ reverse slant	75	75	0134
M	M	15	15	0115	^ circumflex	^ circumflex	76†	7402	0136
N	N	16	16	0116	; semicolon	; semicolon	77	77	0073
O	O	17	17	0117					
						: colon		7404	0072
P	P	20	20	0120		% percent		7404	0045
Q	Q	21	21	0121		` grave accent	74†	7407	0140
R	R	22	22	0122		a		7601	0141
S	S	23	23	0123		b		7602	0142
T	T	24	24	0124		c		7603	0143
U	U	25	25	0125		d		7604	0144
V	V	26	26	0126		e		7605	0145
W	W	27	27	0127		f		7606	0146
X	X	30	30	0130		g		7607	0147
Y	Y	31	31	0131		h		7610	0150
Z	Z	32	32	0132		i		7611	0151
0	0	33	33	0060		j		7612	0152
1	1	34	34	0061		k		7613	0153
2	2	35	35	0062		l		7614	0154
3	3	36	36	0063		m		7615	0155
4	4	37	37	0064		n		7616	0156
						o		7617	0157
5	5	40	40	0065		p		7620	0160
6	6	41	41	0066		q		7621	0161
7	7	42	42	0067		r		7622	0162
8	8	43	43	0070		s		7623	0163
9	9	44	44	0071		t		7624	0164
+ plus	+ plus	45	45	0053		u		7625	0165
- dash	- dash	46	46	0055		v		7626	0166
* asterisk	* asterisk	47	47	0052		w		7627	0167
/ slant	/ slant	50	50	0057		x		7630	0170
(opening parenthesis	(opening parenthesis	51	51	0050		y		7631	0171
) closing parenthesis	) closing parenthesis	52	52	0051		z		7632	0172
$ dollar sign	$ dollar sign	53	53	0044		{ opening brace	61†	7633	0173
= equal	= equal	54	54	0075		| vertical line	75†	7634	0174
space	space	55	55	0040		} closing brace	62†	7635	0175
, comma	, comma	56	56	0054		~ tilde	76†	7636	0176
. period	. period	57	57	0056		DEL		7637	0177

†The interpretation of this character or code may depend on its context. Refer to Character Set Anomalies elsewhere in this appendix.

Table A-1. Character Sets for Interactive Jobs (Sheet 2 of 2)

ASCII Graphic (64 Character)	ASCII Character (128 Character)	6-Bit Display Code	6/12-Bit Display Code	7-Bit ASCII Code	ASCII Graphic (64 Character)	ASCII Character (128 Character)	6-Bit Display Code	6/12-Bit Display Code	7-Bit ASCII Code
	NUL		7640	4000		DLE		7660	0020
	SOH		7641	0001		DC1		7661	0021
	STX		7642	0002		DC2		7662	0022
	ETX		7643	0003		DC3		7663	0023
	EOT		7644	0004		DC4		7664	0024
	ENQ		7645	0005		NAK		7665	0025
	ACK		7646	0006		SYN		7666	0026
	BEL		7647	0007		ETB		7667	0027
	BS		7650	0010		CAN		7670	0030
	HT		7651	0011		EM		7671	0031
	LF		7652	0012		SUB		7672	0032
	VT		7653	0013		ESC		7673	0033
	FF		7654	0014		FS		7674	0034
	CR		7655	0015		GS		7675	0035
	SO		7656	0016		RS		7676	0036
	SI		7657	0017		US†		7677	0037

†Reserved for network use. Refer to Character Set Tables in this appendix.

Table A-2. Character Sets for Batch Jobs (Sheet 1 of 3)

CDC Graphic (64 Character)	ASCII Graphic (64 Character)	ASCII Graphic (95 Character)	6-Bit Display Code	6/12-Bit Display Code	7-Bit ASCII Code	Punch Code	
						026	029
: colon†	: colon†		00†			8-2	8-2
Display code 00 is undefined at sites using the 63-character set.							
A	A	A	01	01	0101	12-1	12-1
B	B	B	02	02	0102	12-2	12-2
C	C	C	03	03	0103	12-3	12-3
D	D	D	04	04	0104	12-4	12-4
E	E	E	05	05	0105	12-5	12-5
F	F	F	06	06	0106	12-6	12-6
G	G	G	07	07	0107	12-7	12-7
H	H	H	10	10	0110	12-8	12-8
I	I	I	11	11	0111	12-9	12-9
J	J	J	12	12	0112	11-1	11-1
K	K	K	13	13	0113	11-2	11-2
L	L	L	14	14	0114	11-3	11-3
M	M	M	15	15	0115	11-4	11-4
N	N	N	16	16	0116	11-5	11-5
O	O	O	17	17	0117	11-6	11-6
P	P	P	20	20	0120	11-7	11-7
Q	Q	Q	21	21	0121	11-8	11-8
R	R	R	22	22	0122	11-9	11-9
S	S	S	23	23	0123	0-2	0-2
T	T	T	24	24	0124	0-3	0-3
U	U	U	25	25	0125	0-4	0-4
V	V	V	26	26	0126	0-5	0-5
W	W	W	27	27	0127	0-6	0-6
X	X	X	30	30	0130	0-7	0-7
Y	Y	Y	31	31	0131	0-8	0-8
Z	Z	Z	32	32	0132	0-9	0-9
0	0	0	33	33	0060	0	0
1	1	1	34	34	0061	1	1
2	2	2	35	35	0062	2	2
3	3	3	36	36	0063	3	3
4	4	4	37	37	0064	4	4
5	5	5	40	40	0065	5	5
6	6	6	41	41	0066	6	6
7	7	7	42	42	0067	7	7
8	8	8	43	43	0070	8	8
9	9	9	44	44	0071	9	9
+ plus	+ plus	+ plus	45	45	0053	12	12-8-6
- dash	- dash	- dash	46	46	0055	11	11
* asterisk	* asterisk	* asterisk	47	47	0052	11-8-4	11-8-4

†The interpretation of this character or code may depend on its context. Refer to Character Set Anomalies elsewhere in this appendix.

Table A-2. Character Sets for Batch Jobs (Sheet 2 of 3)

CDC Graphic (64 Character)	ASCII Graphic (64 Character)	ASCII Graphic (95 Character)	6-Bit Display Code	6/12-Bit Display Code	7-Bit ASCII Code	Punch Code	
						026	029
/ slant	/ slant	/ slant	50	50	0057	0-1	0-1
(opening parenthesis	(opening parenthesis	(opening parenthesis	51	51	0050	0-8-4	12-8-5
) closing parenthesis	) closing parenthesis	) closing parenthesis	52	52	0051	12-8-4	11-8-5
$ dollar sign	$ dollar sign	$ dollar sign	53	53	0044	11-8-3	11-8-3
= equal	= equal	= equal	54	54	0075	8-3	8-6
space	space	space	55	55	0040	no punch	no punch
, comma	, comma	, comma	56	56	0054	0-8-3	0-8-3
. period	. period	. period	57	57	0056	12-8-3	12-8-3
≡ equivalence	# number sign	# number sign	60	60	0043	0-8-6	8-3
[opening bracket	[opening bracket	[opening bracket	61	61	0133	8-7	12-8-2†
] closing bracket	] closing bracket	] closing bracket	62	62	0135	0-8-2	11-8-2†
% percent sign†	% percent sign†	% percent sign†	63†	63†	0045	8-6	0-8-4
: colon	: colon	: colon	63	63	0072	8-2	8-2
≠ not equal	" quote	" quote	64	64	0042	8-4	8-7
↱ concatenation	_ underline	_ underline	65	65	0137	0-8-5	0-8-5
∨ logical OR	! exclamation point	! exclamation point	66	66	0041	11-0	12-8-7
∧ logical AND	& ampersand	& ampersand	67	67	0046	0-8-7	12
↑ superscript	' apostrophe	' apostrophe	70	70	0047	11-8-5	8-5
↓ subscript	? question mark	? question mark	71	71	0077	11-8-6	0-8-7
< less than	< less than	< less than	72	72	0074	12-0	12-8-4
> greater than	> greater than	> greater than	73	73	0076	11-8-7	0-8-6
≤ less or equal	@ commercial at		74			8-5	8-4
≥ greater or equal	\ reverse slant	\ reverse slant	75	75	0134	12-8-5	0-8-2
¬ logical NOT	^ circumflex		76			12-8-6	11-8-7
; semicolon	; semicolon	; semicolon	77	77	0073	12-8-7	11-8-6
		@ commercial at	74†	7401	0100		
		^ circumflex	76†	7402	0136		
		: colon†		7404†	0072		
		% percent sign		7404	0045		
		` grave accent	74†	7407	0140		
		a		7601	0141		
		b		7602	0142		
		c		7603	0143		
		d		7604	0144		
		e		7605	0145		
		f		7606	0146		
		g		7607	0147		

†The interpretation of this character or code may depend on its context. Refer to Character Set Anomalies elsewhere in this appendix.

Table A-2. Character Sets for Batch Jobs (Sheet 3 of 3)

CDC Graphic (64 Character)	ASCII Graphic (64 Character)	ASCII Graphic (95 Character)	6-Bit Display Code	6/12-Bit Display Code	7-Bit ASCII Code	Punch Code	
						026	029
		h		7610	0150		
		i		7611	0151		
		j		7612	0152		
		k		7613	0153		
		l		7614	0154		
		m		7615	0155		
		n		7616	0156		
		o		7617	0157		
		p		7620	0160		
		q		7621	0161		
		r		7622	0162		
		s		7623	0163		
		t		7624	0164		
		u		7625	0165		
		v		7626	0166		
		w		7627	0167		
		x		7630	0170		
		y		7631	0171		
		z		7632	0172		
		{ opening brace	61†	7633	0173		
		| vertical line	75†	7634	0174		
		} closing brace	62†	7635	0175		
		~ tilde	76†	7636	0176		

† The interpretation of this character or code may depend on its context. Refer to Character Set Anomalies elsewhere in this appendix.

Table A-3. ASCII to 6/12-Bit Display Code Conversion (Sheet 1 of 2)

ASCII Character (128 Character)	7-Bit ASCII Code		6/12-Bit Display Code	ASCII Character (128 Character)	7-Bit ASCII Code		6/12-Bit Display Code
	Octal	Hexadecimal			Octal	Hexadecimal	
NUL	4000	00	7640	0	0060	30	33
SOH	0001	01	7641	1	0061	31	34
STX	0002	02	7642	2	0062	32	35
ETX	0003	03	7643	3	0063	33	36
EOT	0004	04	7644	4	0064	34	37
ENQ	0005	05	7645	5	0065	35	40
ACK	0006	06	7646	6	0066	36	41
BEL	0007	07	7647	7	0067	37	42
BS	0010	08	7650	8	0070	38	43
HT	0011	09	7651	9	0071	39	44
LF	0012	0A	7652	: colon††	0072	3A	7404††
VT	0013	0B	7653	: colon	0072	3A	63
FF	0014	0C	7654	; semicolon	0073	3B	77
CR	0015	0D	7655	< less than	0074	3C	72
SO	0016	0E	7656	= equal	0075	3D	54
SI	0017	0F	7657	> greater than	0076	3E	73
				? question mark	0077	3F	71
DLE	0020	10	7660				
DC1	0021	11	7661	@ commercial at	0100	40	7401
DC2	0022	12	7662	A	0101	41	01
DC3	0023	13	7663	B	0102	42	02
DC4	0024	14	7664	C	0103	43	03
NAK	0025	15	7665	D	0104	44	04
SYN	0026	16	7666	E	0105	45	05
ETB	0027	17	7667	F	0106	46	06
				G	0107	47	07
CAN	0030	18	7670				
EM	0031	19	7671	H	0110	48	10
SUB	0032	1A	7672	I	0111	49	11
ESC	0033	1B	7673	J	0112	4A	12
FS	0034	1C	7674	K	0113	4B	13
GS	0035	1D	7675	L	0114	4C	14
RS	0036	1E	7676	M	0115	4D	15
US†	0037	1F	7677†	N	0116	4E	16
				O	0117	4F	17
space	0040	20	55				
! exclamation point	0041	21	66	P	0120	50	20
" quote	0042	22	64	Q	0121	51	21
# number sign	0043	23	60	R	0122	52	22
$ dollar sign	0044	24	53	S	0123	53	23
% percent sign††	0045	25	63††	T	0124	54	24
% percent sign	0045	25	7404	U	0125	55	25
& ampersand	0046	26	67	V	0126	56	26
' apostrophe	0047	27	70	W	0127	57	27
(opening parenthesis	0050	28	51	X	0130	58	30
) closing parenthesis	0051	29	52	Y	0131	59	31
* asterisk	0052	2A	47	Z	0132	5A	32
+ plus	0053	2B	45	[opening bracket	0133	5B	61
, comma	0054	2C	56	\ reverse slant	0134	5C	75
- dash	0055	2D	46	] closing bracket	0135	5D	62
. period	0056	2E	57	^ circumflex	0136	5E	7402
/ slant	0057	2F	50	_ underline	0137	5F	65

†Reserved for network use. Refer to Character Set Tables in this appendix.
††The interpretation of this character or code may depend on its context. Refer to Character Set Anomalies in this appendix.

Table A-3. ASCII to 6/12-Bit Display Code Conversion (Sheet 2 of 2)

ASCII Character (128 Character)	7-Bit ASCII Code		6/12-Bit Display Code	ASCII Character (128 Character)	7-Bit ASCII Code		6/12-Bit Display Code
	Octal	Hexadecimal			Octal	Hexadecimal	
` grave accent	0140	60	7407	p	0160	70	7620
a	0141	61	7601	q	0161	71	7621
b	0142	62	7602	r	0162	72	7622
c	0143	63	7603	s	0163	73	7623
d	0144	64	7604	t	0164	74	7624
e	0145	65	7605	u	0165	75	7625
f	0146	66	7606	v	0166	76	7626
g	0147	67	7607	w	0167	77	7627
h	0150	68	7610	x	0170	78	7630
i	0151	69	7611	y	0171	79	7631
j	0152	6A	7612	z	0172	7A	7632
k	0153	6B	7613	{ opening brace	0173	7B	7633
l	0154	6C	7614	| vertical line	0174	7C	7634
m	0155	6D	7615	} closing brace	0175	7D	7635
n	0156	6E	7616	~ tilde	0176	7E	7636
o	0157	6F	7617	DEL	0177	7F	7637

JOBS USING MAGNETIC TAPE

Coded data to be copied from mass storage to magnetic tape is assumed to be represented in display code. NOS converts the data to external BCD code when writing a coded seven-track tape and to ASCII or EBCDIC code (as specified on the tape assignment command) when writing a coded nine-track tape.

Because only 63 characters can be represented in seven-track even parity, one of the 64 display codes is lost in conversion to and from external BCD code. The following shows the differences in conversion depending on the character set (63 or 64) which the system uses. The ASCII character for the specified character code is shown in parentheses. The output arrow shows how the 6-bit display code changes when it is written on tape in external BCD. The input arrow shows how the external BCD code changes when the tape is read and converted to 6-bit display code.

63-Character Set

6-Bit Display Code		External BCD		6-Bit Display Code
00		16 (%)		00
33 (0)	Output →	12 (0)	Input →	33 (0)
63 (:)		12 (0)		33 (0)

64-Character Set

6-Bit Display Code		External BCD		6-Bit Display Code
00 (:)		12 (0)		33 (0)
33 (0)	Output →	12 (0)	Input →	33 (0)
63 (%)		16 (%)		63 (%)

If a lowercase ASCII or EBCDIC code is read from a nine-track coded tape, it is converted to its uppercase 6-bit display code equivalent. To read and write lowercase ASCII or EBCDIC characters, you must use FCOPY to convert the file to 6/12-bit display code.

Tables A-4 and A-5 show the character set conversion for nine-track tapes. Table A-4 lists the conversions to and from the 7-bit ASCII character code and 6-bit display code. Table A-5 lists the conversions between the EBCDIC character code and the 6-bit display code. Table A-6 shows the character set conversions between external BCD and 6-bit display code for seven-track tapes.

Table A-4. Nine-Track ASCII Coded Tape Conversion

7-Bit ASCII				6 Bit Display Code		7-bit ASCII				6-Bit Display Code	
Code (Hex)	Char†	Code (Hex)	Char††	Char	Code (Octal)	Code (Hex)	Char†	Code (Hex)	Char††	Char	Code (Octal)
20	space	00	NUL	space	55	3E	>	1E	RS	>	73
21	!	7D	}	!	66	3F	?	1F	US	?	71
22	"	02	STX	"	64	40	@	60	`	@	74
23	#	03	ETX	#	60	41	A	61	a	A	01
24	$	04	EOT	$	53	42	B	62	b	B	02
25	%	05	ENQ	%	63	43	C	63	c	C	03
25	%	05	ENQ	space†††	55	44	D	64	d	D	04
26	&	06	ACK	&	67	45	E	65	e	E	05
27	'	07	BEL	'	70	46	F	66	f	F	06
28	(	08	BS	(	51	47	G	67	g	G	07
29	)	09	HT	)	52	48	H	68	h	H	10
2A	*	0A	LF	*	47	49	I	69	i	I	11
2B	+	0B	VT	+	45	4A	J	6A	j	J	12
2C	,	0C	FF	,	56	4B	K	6B	k	K	13
2D	-	0D	CR	-	46	4C	L	6C	l	L	14
2E	.	0E	SO	.	57	4D	M	6D	m	M	15
2F	/	0F	SI	/	50	4E	N	6E	n	N	16
30	0	10	DLE	0	33	4F	O	6F	o	O	17
31	1	11	DC1	1	34	50	P	70	p	P	20
32	2	12	DC2	2	35	51	Q	71	q	Q	21
33	3	13	DC3	3	36	52	R	72	r	R	22
34	4	14	DC4	4	37	53	S	73	s	S	23
35	5	15	NAK	5	40	54	T	74	t	T	24
36	6	16	SYN	6	41	55	U	75	u	U	25
37	7	17	ETB	7	42	56	V	76	v	V	26
38	8	18	CAN	8	43	57	W	77	w	W	27
39	9	19	EM	9	44	58	X	78	x	X	30
3A	:	1A	SUB	:	00	59	Y	79	y	Y	31
6-Bit display code 00 is undefined at sites using the 63-character set.						5A	Z	7A	z	Z	32
						5B	[	1C	FS	[	61
3A	:	1A	SUB	:	63	5C	\	7C	|	\	75
3B	;	1B	ESC	;	77	5D	]	01	SOH	]	62
3C	<	7B	{	<	72	5E	^	7E	~	^	76
3D	=	1D	GS	=	54	5F	_	7F	DEL	_	65

†When these characters are copied from/or to a tape, the characters remain the same but the codes change from one code set to the other.

††These characters do not exist in 6-bit display code. Therefore, when the characters are copied from a tape, each 7-bit ASCII character is changed to an alternate 6-bit display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 61_{16}, from tape, it writes an uppercase A, 01_{8}.

†††A 6-bit display code space always translates to a 7-bit ASCII space.

Table A-5. Nine-Track EBCDIC Coded Tape Conversion

EBCDIC				6-Bit Display Code		EBCDIC				6-Bit Display Code	
Code (Hex)	Char†	Code (Hex)	Char††	Char††	Code (Octal)	Code (Hex)	Char†	Code (Hex)	Char††	Char	Code (Octal)
40	space	00	NUL	space	55	C4	D	84	d	D	04
4A	¢	1C	IFS	[	61	C5	E	85	e	E	05
4B	.	0E	SO	.	57	C6	F	86	f	F	06
4C	<	C0	{	<	72	C7	G	87	g	G	07
4D	(	16	BS	(	51	C8	H	88	h	H	10
4E	+	0B	VT	+	45	C9	I	89	i	I	11
4F	|	D0	}	!	66	D1	J	91	j	J	12
50	&	2E	ACK	&	67	D2	K	92	k	K	13
5A	!	01	SOH	]	62	D3	L	93	l	L	14
5B	$	37	EOT	$	53	D4	M	94	m	M	15
5C	*	25	LF	*	47	D5	N	95	n	N	16
5D	)	05	HT	)	52	D6	O	96	o	O	17
5E	;	27	ESC	;	77	D7	P	97	p	P	20
5F	¬	A1	~	^	76	D8	Q	98	q	Q	21
60	-	0D	CR	-	46	D9	R	99	r	R	22
61	/	0F	SI	/	50	E0	\	6A	|	\	75
6B	,	0C	FF	,	56	E2	S	A2	s	S	23
6C	%	2D	ENQ	%	63	E3	T	A3	t	T	24
6C	%	2D	ENQ	space†††	55	E4	U	A4	u	U	25
6D	_	07	DEL	_	65	E5	V	A5	v	V	26
6E	>	1E	IRS	>	73	E6	W	A6	w	W	27
6F	?	1F	IUS	?	71	E7	X	A7	x	X	30
7A	:	3F	SUB	:	00	E8	Y	A8	y	Y	31
6-Bit display code 00 is undefined at sites using the 63-character set.						E9	Z	A9	z	Z	32
						F0	0	10	DLE	0	33
7A	:	3F	SUB	:	63	F1	1	11	DC1	1	34
7B	#	03	ETX	#	60	F2	2	12	DC2	2	35
7C	@	79	\	@	74	F3	3	13	TM	3	36
7D	'	2F	BEL	'	70	F4	4	3C	DC4	4	37
7E	=	1D	IGS	=	54	F5	5	3D	NAK	5	40
7F	"	02	STX	"	64	F6	6	32	SYN	6	41
C1	A	81	a	A	01	F7	7	26	ETB	7	42
C2	B	82	b	B	02	F8	8	18	CAN	8	43
C3	C	83	c	C	03	F9	9	19	EM	9	44

†When these characters are copied from/or to a tape, the characters remain the same (except EBCDIC codes 4A, 4F, 5A, and 5F) but the codes change from one code set to the other.

††These characters do not exist in 6-bit display code. Therefore, when the characters are copied from a tape, each EBCDIC character is changed to an alternate 6-bit display code character. The corresponding codes are also changed. Example: When the system copies a lowercase a, 81_{16}, from tape, it writes an uppercase A, 01_8.

†††All EBCDIC codes not listed translate to 6-bit display code 55_8 (space). A 6-bit display code space always translates to an EBCDIC space.

Table A-6. Seven-Track Coded Tape Conversions

External BCD	ASCII Character	6-Bit Display Code (Octal)	External BCD	ASCII Character	6-Bit Display Code (Octal)
01	1	34	40	-	46
02	2	35	41	J	12
03	3	36	42	K	13
04	4	37	43	L	14
05	5	40	44	M	15
06	6	41	45	N	16
07	7	42	46	O	17
10	8	43	47	P	20
11	9	44	50	Q	21
12†	0	33	51	R	22
13	=	54	52	!	66
14	"	64	53	$	53
15	@	74	54	*	47
16†	%	63	55	'	70
17	[	61	56	?	71
20	space	55	57	>	73
21	/	50	60	+	45
22	S	23	61	A	01
23	T	24	62	B	02
24	U	25	63	C	03
25	V	26	64	D	04
26	W	27	65	E	05
27	X	30	66	F	06
30	Y	31	67	G	07
31	Z	32	70	H	10
32	]	62	71	I	11
33	,	56	72	<	72
34	(	51	73	.	57
35		65	74	)	52
36	#	60	75	\	75
37	&	67	76	^	76
			77	;	77

†As explained previously in this section, conversion of these codes depends on whether the tape is being read or written.

GLOSSARY B

Abort

To terminate a program or job when an error condition (hardware or software) exists from which the program or computer cannot recover.

Absolute Address

An address within a program, determined by the loader, that is fixed in relation to the first word of the program.

Absolute Load

A load of an overlay, segment, or other absolute program. No relocation or satisfaction of externals is needed because this was done when the absolute program was generated.

Acoustic Coupler

A device in which you place a telephone receiver to connect a terminal to a telephone line. The telephone line is in turn connected to the system. After dialing the system, you place the telephone receiver in the rubber cups of the acoustic coupler, ensuring that the telephone-cord end of the receiver is at the correct end (marked on the coupler). An acoustic coupler is either built into the terminal (common on portable terminals) or a separate part (common on nonportable terminals). Contrast with Data Set.

Alphanumeric Characters

The letters of the alphabet (A through Z) and the digits (0 through 9).

Application

A program resident in a host computer that provides an information storage, retrieval, and/or processing service to a remote user via the network. IAF is an example of an application.

ASCII

American National Standard Code for Information Interchange. The standard character set and code used for information interchange between systems.

ASCII Mode

Mode that uses the entire ASCII character set. ASCII stands for American Standard Code for Information Interchange. It is a code that can represent 128 characters, including uppercase and lowercase letters, numbers, and special characters. In ASCII mode, you can enter uppercase and lowercase letters and the system will not translate them all to uppercase as it does in normal mode. Contrast with Normal Mode.

Auxiliary Device

Mass storage permanent file device used to supplement storage provided by the normal permanent file devices associated with the system (refer to Family Device).

BASIC

1. Beginner's All-Purpose Symbolic Instruction Code, an elementary programming language.

2. The subsystem that uses the BASIC compiler.

Basic Load

A load operation in which all of the object code is loaded into memory at the same time.

Beginning-of-Information (BOI)

A file boundary that marks the beginning of a file. You can reference the BOI by name. For a labeled tape file on a single volume, the HDR1 represents BOI. If the file is a multivolume (multireel) tape file, BOI is the HDR1 label of the first volume. If the tape file is an ANSI multifile file, BOI is the HDR1 associated with the beginning of the multifile.

Bit

An abbreviation of binary digit. It is a single digit, 0 or 1, in a binary number. Bit is also used to represent the smallest unit of information. A central memory word (one storage location) contains 60 bits.

Block

Blocking is the grouping of user records for efficiency in transfer between memory and storage devices. For magnetic tapes, it is the information between interrecord gaps.

BOI

Refer to Beginning-of-Information.

Byte

A group of 12 bits. Five bytes compose a 60-bit central memory word. Bytes are numbered 0 to 4 from the left.

Catalog

A collection of information about the permanent files associated with a particular user name. Every user who saves permanent files has a permanent file catalog. Each time you create, modify, or purge a permanent file, the system updates your catalog accordingly. You can list the names of your permanent files and other information about them using the CATLIST command.

Character

Any alphabetic, numeric, or special symbol that can be encoded. This term applies to the graphic characters for an input or output device, and to uniquely encoded control characters used by a terminal.

Command

A sequence of words and characters that call a system routine to perform a job step. The command must conform to format specifications. You can usually place a comment after the command terminator.

Command Record

The first, and possibly only, record of a job consisting of command images that start with the Job command and end with the first EOR, EOF, or EOI.

Compile

To translate a program from a higher level programming language (for example, FORTRAN or BASIC) into machine instructions called object code.

Control Key

A terminal key labeled CONTROL, CTRL, CNTL, CNTRL, or something similar. Many terminals require the control key to be pressed and held while another key is pressed for the cancel line sequence, interruption sequence, or termination sequence.

Control Point

The portion of central memory that is assigned to a job. When a job is allocated a portion of central memory, it becomes eligible for assignment to the central processor for execution.

Control Statement

Refer to Command.

CYBER Loader

The utility that prepares programs for execution by placing program instruction and data blocks in central memory.

CYBER Record Manager (CRM)

A software product that allows a variety of record types, blocking types, and file organizations to be created and accessed. Products such as COBOL 5, FORTRAN Extended 4, FORTRAN 5, Sort/Merge 4, ALGOL 5, and DMS 170 use CRM to manage execution time input/output. Neither the input/output of the operating system nor that of most of the system utilities such as COPY or SKIPF is implemented through CYBER Record Manager. All CYBER Record Manager file processing requests ultimately pass through the operating system input/output routines.

Dayfile

A chronological file created during job execution which forms a permanent accounting and job history record. Dayfile messages are generated by operator action or when commands are processed. A copy of the dayfile is printed with the output for each noninteractive job. You must explicitly request it in an interactive job.

Default

A system-supplied option used when you do not supply the option.

Device

Peripheral equipment.

Direct Access File

A NOS permanent mass storage file that can be attached to your job. All changes to this file are made on the file itself rather than on a temporary copy of the file (contrast with Indirect Access File).

Directive

An instruction subordinate to a system command.

Display Code

A 6-bit-character code set used to represent alphanumeric and special characters.

Duplex

Mode of transmission on a communication line. You need be concerned about duplex mode only to the extent of checking that the duplex switch on your terminal is set correctly for the line being used. Although the switch is normally set to HALF, personnel at your computer center can tell you the correct setting for your terminal. If you enter data but nothing is printed, check the duplex switch on your terminal; it is probably set incorrectly.

EBCDIC

Extended Binary Coded Decimal Interchange Code. An 8-bit code which represents a set of 256 characters.

Empty PRU/Record

A PRU that contains no user data. Same as zero-length PRU.

End-of-File (EOF)

A boundary within a sequential file, but not necessarily the end of a file that can be referenced by name. The actual end of a named file is defined by EOI. For labeled tape, EOF and EOI (denoted by the EOF1 label) are the same. For multifile tape files, EOF and EOI do not correspond. In the product set manuals, an end-of-file is also referred to as an end-of-partition.

End-of-Information (EOI)

The end of data on a file. Information appearing after this point is not considered part of file data. In card decks, it is a card with a 6/7/8/9 multiple punch in column 1. On mass storage devices, it is the position of the last written data. On labeled tape, it is the EOF1 label. CYBER Record Manager defines end-of-information in terms of file residency and organization.

End-of-Record (EOR)

An indicator that marks the end of a logical record. Also referred to as end-of-section.

End-of-Tape

A reflective strip near the end of a magnetic tape. It is used to signal termination of operations on a particular tape volume. At least 5.5 metres (18 feet) of tape must follow this marker.

Entry Point

A location within a program or procedure that can be referenced from other programs. Each entry point has a unique name with which it is associated.

EOF

Refer to End-of-File.

EOI

Refer to End-of-Information.

EOR

Refer to End-of-Record.

EOT

Refer to End-of-Tape.

Error Flag

A character or bit that signals the occurrence or presence of an error.

Exchange Package

A table that contains information used during job execution. It is printed as part of the output when a job aborts.

Executable Object Code

Machine language instructions that can be executed directly by the machine. A compiled program is composed of executable object code.

Execute-Only File

A file that can be executed but cannot be written to or read from. Execute-only permission is established by a parameter on the DEFINE or SAVE command.

External Reference

A reference in one compiled program to an entry point in another program.

Family Device

Mass storage permanent file device associated with a specific system. A family may consist of from 1 through 63 logical devices. Normally, a system runs with one family of permanent file devices available. However, additional families may be introduced during normal operation. This enables users associated with the additional families to access their permanent files on more than one system.

Family Name

Name of the permanent file storage device or set of devices on which all of your permanent files are stored. When you request a permanent file, the system looks for it on this family (group) of devices. Usually a system has only one family of permanent file devices, but it is possible to have alternative families in the system. You specify which family you are using when you log in. Your family name is given to you by your employer, instructor, or computer center personnel.

Field Length (FL)

The central memory and extended memory assigned an executing job.

File

1. A set of information that begins at beginning-of-information (BOI), ends at end-of-information (EOI), and is referenced by a local file name.

2. That portion of a multifile file terminated by an end-of-file (EOF).

3. Data recorded on a magnetic tape beginning after an HDR1 label and ending before an EOF1 label.

NOS commands requiring a parameter that is a file name refer to definition 1.

Commands requiring a parameter that specifies the number of files refer to definition 2.

Definition 3 applies only to labeled magnetic tapes.

File Set

One or more tape files referred to by the set identifier on a tape assignment command. A file set may consist of:

1. One file recorded on a single volume.

2. More than one file recorded on a single volume.

3. One file recorded on more than one volume.

4. More than one file recorded on more than one volume.

All files within a file set have the same set identifier in their HDR1 labels.

Flag

A character or bit that signals the occurrence or presence of a particular condition.

FORTRAN

1. Formula Translation. A high-level language consisting of symbols and statements that can be used to create a program closely resembling mathematical notation.

2. The subsystem that uses the FORTRAN 5 compiler.

FSE (Full Screen Editor)

Refer to Full Screen Editor.

FTNTS

The subsystem that uses the FORTRAN Extended 4 compiler.

Full Screen Editor

A text editor that gives you two methods of editing: line editing and screen editing. Line editing can be used on any kind of terminal and allows you to edit a line by typing a directive and specifying the appropriate line number. Screen editing can be used only on certain display terminals and allows you to move the cursor about freely on the screen and use special function keys on the keyboard to perform editing operations.

Global Library Set

A set of libraries you specify on the LIBRARY command. The system conditionally searches these libraries when it attempts to process a command.

Hardwired Terminal

A terminal that is always connected to the system. To begin a terminal session on a hardwired terminal, all you need to do is identify the terminal by pressing the proper key(s), and the login sequence begins (refer to Entering the System in section 2).

Host

A basic component of a computer system, consisting of a mainframe, its operating system (NOS), and applications.

Host Path

One or more entry points to a host from the network.

IAF

Refer to Interactive Facility.

Indirect Access File

A NOS permanent file that you access by making a temporary copy of the file (GET or OLD command). You create or alter it by saving or substituting the contents of an existing temporary file (REPLACE or SAVE command).

Interactive Facility (IAF)

An application program that provides a terminal operator with interactive processing capability. The interactive facility makes terminal input/output and file input/output appear the same to an executing program.

Interruption Sequence

The character or sequence of characters that interrupts and suspends a system operation (for example, listing a program or running a program). It may differ on the terminal you're using, but usually you hold down the control key and press the P key. Then release both keys and press the carriage return. To resume the operation, you press the carriage return.

Job

A set of commands and the data and directives used by those commands. A batch job must begin with the Job and USER commands. An interactive job is all activity associated with a terminal session from login to logout.

Job Sequence Name (JSN)

The unique, system-defined name assigned to every executing job. The JSN is a string of four alphabetic characters.

Job Step

An individual command or loader sequence. A group of job steps forms a job stream or command record.

Label

A block at the beginning or end of a magnetic tape volume or file, which serves toidentify and/or delimit that volume or file.

Labeled Tape

Refer to Standard Labeled Tape.

Library

1. A collection of programs or routines.
2. A file containing records that are accessed individually.
3. A file searched by CYBER Loader for entry points referenced by a program.
4. A file containing compressed records in Modify or Update format.

Library File

A read-only file that can be accessed by several users simultaneously or a file you specify on a LIBRARY command.

Library Set

An ordered set of libraries that the loader searches to satisfy external references. Two library sets can be established: a global library set and a local library set. The global set is searched first, followed by the local set, followed by the system library SYSLIB.

Line Mode

A mode of job interaction in which the basic unit of interaction is a line of data. Contrast with Screen Mode.

Load Map

A loader printout showing how memory was allocated by the loader during a load operation.

Loader Command

A command that provides processing instructions to the loader.

Loading Sequence

One or more consecutive loader commands processed by the loader as a unit. A loading sequence can be a single name call command, or it can consist of multiple loader commands, such as LOAD and LDSET, that are terminated by NOGO, EXECUTE, or a name call command.

Local File

Any file that is currently associated with a job. Local files include all temporary files and attached direct access files.

Local File Name

The file name assigned to a file while it is local (associated with) a job.

Local Library Set

An ordered set of libraries defined by LDSET (LIB=) commands and object directives, and used for a single loading sequence.

Locked File

A file on which you cannot write.

Logical Device

One or more physical disk units known to the system as a single device.

Logical Record

A data grouping that consists of zero or more PRUs and ends with a short PRU or a zero-length PRU.

Login

Process during which you identify yourself to the system as an authorized user. To identify yourself, you enter certain responses to prompts from the system. These responses are given to you initially by your employer, instructor, or computer center personnel. They include family name, user name, and password.

Logout

Process by which you end a terminal session. You type BYE.

Mass Storage

Magnetic disk or extended memory that can be acessed randomly as well as sequentially.

Master Device

Mass storage device that contains your permanent file catalog entries; all your indirect access files; and all, part, or none of your direct access files.

Menu Procedure

An interactive procedure that displays a list of options from which you make a selection for execution.

Multifile File

A file containing more than one logical file. It begins at BOI and ends at EOI. On a labeled tape, a multifile file is delimited by corresponding HDR1 and EOF1 labels.

Multifile Set

A tape file set having more than one tape file.

Multihost Arrangement

A system with more than one host available for use through the same network. Contrast with Single-Host Arrangement.

Name Call Command

A local file that you can execute by simply specifying its name.

Network Control Character

A character which, when entered at the beginning of a line, signals to the network that a terminal definition follows.

Network

A data and message switching and routing system used to provide communication between terminals and applications.

Nonallocatable Device

A device such as a magnetic tape unit, card reader, card punch, or line printer which can be used only by one job at a time.

Normal Mode

Mode in which the system translates any lowercase letters that you enter to uppercase letters. Contrast with ASCII Mode.

NOS

Network Operating System.

Object Code

Executable machine language instructions. An object code program need not be recompiled each time it is executed.

Object Module

A compiled program suitable for input to the loader.

Order-Dependent

Used to describe items which must appear in a specific order.

Order-Independent

Used to describe items which need not appear in any specific order. Parameters, particularly those with keywords, may be order-independent.

Origin Type

A job attribute that indicates how a job entered the system. The four origin types are interactive origin, batch origin, remote batch origin, and system origin.

OVCAP

A special capsule designed for use with overlays. An OVCAP is analagous to a primary overlay in that it must be called into memory by a main overlay, and it can reference entry points and common blocks in the main overlay.

Overlay

One or more relocatable program modules that were relocated and linked together into a single absolute program.

Parameter

A value that follows a command name or function name and that alters the behavior of the command or function.

Parity

Mode indicating a method of error detection. You need be concerned about parity mode only to the extent of checking that the parity switch on your terminal is set correctly. Personnel at your computer center can tell you the correct setting for your terminal. Parity is either even, odd, or disabled (not used). When used, a special bit (the parity bit) is set or cleared in the string of binary digits, so that the sum of the digits set, including the partity bit, is always even or always odd. When data is read, the sum is checked and, if it doesn't agree with the parity setting (even or odd), the system knows the data is incorrect.

Password

1. A system access word that must be used in addition to the user name at login or on the USER command.

2. A file access word that defines access to a particular file by other users.

Permanent File

A file stored on mass storage. This file is cataloged by the system so that its location and identification are always known to the system. Permanent files cannot be purged accidentally during normal system operation. They are protected by the system from unauthorized access according to privacy controls specified by the file owner.

Permanent File Device

Mass storage defined by your site to hold permanent files.

Permanent File Family

Permanent files which reside on the family devices of a specific system.

Physical Record Unit (PRU)

The amount of information transmitted by a single physical operation of a specified device. For mass storage files, a PRU is 64 central memory words (640 characters); for magnetic tape files, the size of the PRU depends upon the tape format. A PRU that is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU.

Prefix Character

A character that has a special significance to a program or the operating system and is used in front of a string of characters.

Primary File

A temporary file created with the OLD, NEW, LIB (interactive jobs only), or PRIMARY command. The primary file is assumed to be the file on which most system operations are performed unless another file is specified. There can be only one primary file associated with a user's job.

Procedure

A user-defined set of instructions that can be referenced by name. The instructions consist of procedure directives and system commands.

Procedure File

A file containing one or more procedures.

Prologue

A procedure the system executes every time you login. A prologue is executed after you enter your user name and password, but before you are allowed to enter any system commands. You can create your own user prologue. Your site administration and project management can also create prologues associated with your user name.

PRU

Refer to Physical Record Unit.

Random Access

Access method by which any record in a file can be accessed at any time. Applies only to mass storage files with an organization other than sequential. Contrast with Sequential Access.

Random Access File

A file whose records are accessed through a directory containing the address of each record.

Read-Only Permission

If you have read-only permission on a file, you can read and execute the file. You cannot write, modify, append, or purge the file.

Record

A unit of information. In CYBER Record Manager and its language processors, a record is a unit of information produced by a single read or write request.

A NOS record is a subdivision of a file. It is the smallest subdivision that NOS recognizes. A NOS record can consist of a group of CRM records. Refer to Logical Record.

Reference Address (RA)

The absolute machine address in central memory of the first word of a loaded program.

Relative Address

An address within a program unit that is relative to the beginning of that program. The loader converts relative addresses to absolute addresses.

Satisfying External References

The process of searching one or more libraries and loading programs that contain entry points matching external references that are currently unsatisfied.

Screen Mode

A mode of job interaction in which the basic unit of interaction is a screen of data. Contrast with Line Mode.

Secured System

A system with mandatory security that protects information by restricting access to files. Restriction is based on access levels and categories.

Segment

An absolute subdivision of a segmented program that is automatically called into memory as needed (except for the root segment). Different segments can occupy the same memory locations at different times during job execution.

Segmentation

Dividing a program into sections called segments that can occupy the same storage locations at different times. The root segment must be in memory throughout program execution; all other segments are loaded dynamically during program execution.

Sequence Number

1. Number at the beginning of each line of a file.

2. For tape labels, number indicating position of a file within a file set.

Sequential Access

A method in which only the record located at the current file position can be accessed. Refer to Random Access.

Sequential (SQ) File

A file in which records are accessed in the order in which they occur. Any file can be accessed sequentially. Sequential files must be accessed sequentially because no key or address is associated with each record in the files.

Service Class

A job classification based on a job's origin type or job's status. It determines how a job is serviced as it flows through the system.

Single-Host Arrangement

A system in which only one host is available for use through the network, although you may have access to another host through loosely coupled network (LCN). Contrast with Multihost Arrangement.

Source Code

Code input to the computer for later translation into executable machine language instructions (object code).

SRU

Refer to System Resource Unit.

Standard Labeled Tape

A tape with labels conforming to American National Standard Magnetic Tape Labels for Information Interchange X3.27-1969. Also called a system-labeled tape.

String

A sequence of characters.

Submit File

A file containing a job that enters the system for processing as a result of the SUBMIT command. The file is formatted like a batch job. However, it can contain directives that aid in structuring the file.

System

Software and hardware that control the execution of computer program and provide scheduling, error detection, input/output control, accounting, compilation, storage assignment, and related services.

System File

A file that can be accessed only by a system program.

System Library

A library that is installed as part of the operating system. System library names are maintained in a system table and can be used in the library set if the library name is declared in either a LIBRARY or LDSET(LIB=libname) command. (A system library does not have to be attached because it is not a permanent file in the usual sense.)

System Resource Unit (SRU)

A unit of measurement of system use. The number of SRUs includes the central processor time, memory use, and input/output activity.

Tape Format

A parameter that specifies the internal recording format of a magnetic tape.

Tape Mark

A delimiter written on tapes under operating system control to separate label groups, files, and/or labels. Interpretation depends on the tape format.

Temporary File

A file at the terminal that disappears when you either log out or enter the NEW or OLD command without /ND. You create a temporary file either by entering information in a primary file, retrieving an indirect access permanent file (thus making a temporary copy at your terminal), or executing a program that creates a file for results.

Terminal Definition

Two characters that can be entered to instruct the network to change the characteristics of the terminal.

Terminal Name

A network-supplied name used to identify the terminal to the local operator.

Terminal Session

Period between the time you physically connect the terminal to the system during login to the time you log out.

Termination Sequence

The character or sequence of characters that terminates a system operation that is currently in progress (for example, listing a program or running a program). It may differ on the terminal your're using, but usually you hold down the control key and press the T key. Them release both keys and press the carriage return.

Unsatisfied External Reference

An external reference for which no matching entry point was found. The unsatisfied external reference is filled with an address that causes the program to abort if the given instruction is executed.

User Break 1 Sequence

The character or sequence of characters that causes an executing program to be interrupted (also called the interruption sequence).

User Break 2 Sequence

The character or sequence of characters that causes an executing program to be terminated (also called the termination sequence).

User Index

A number the system assigns each user name and uses for internal record keeping.

User Index Hash

A string of four alphabetic characters derived from your user index that serves as the default user job name (UJN) for interactive jobs. It appears on the banner page/card of all line printer and card punch output. You can ascertain your user index hash with the ENQUIRE command.

User Job Name (UJN)

In general, the user-defined name for an executing job or queued file. If you fail to specify a UJN in interactive jobs, the system uses your user index hash. The UJN must be a string of seven or fewer alphanumeric characters. Unlike the job sequence name (JSN), the UJN for jobs is not always unique.

User Library

A library that exists as a local file associated with the job. User libraries can be used in the library set if the library is either created by or associated with a job and declared in either a LIBRARY or LDSET,LIB=libname command.

User Name

A one- through seven-character name that identifies the user to the system and for which there is assigned a user index that allows access to permanent files.

User Number

Refer to User Name.

Validation File

File containing validation information for all users (user names, passwords, resources allowed, and so on).

Volume

A reel of magnetic tape. A given file can encompass more than one volume.

Volume Serial Number (VSN)

A one- through six-character identifier that identifies the volume of magnetic tape for the operator.

VSN

Refer to Volume Serial Number.

Write Interlock

Ensures that only one person at a time can attach a direct access file in write mode. A direct access file that is attached in write mode is in write interlock.

Write Ring

A circular device inserted into a tape reel indicating to the tape unit that it can write on that reel. NOS checks for the presence of a write ring when assigning the tape if you request it.

6/7/8/9 Multipunch

A card with the characters 6, 7, 8, and 9 multipunched in column 1. Signifies a card deck EOI.

6/7/9 Multipunch

A card with the characters 6, 7, and 9 multipunched in column 1. Signifies a card deck EOF.

7/8/9 Multipunch

A card with the characters 7, 8, and 9 multipunched in column 1. Signifies a card deck EOR.

ACCESSING A HOST C

This appendix describes the steps used to access a host and to use CDC network software. The login procedure can be divided into eight steps.

- Gathering information.
- Setting up your terminal.
- Connecting your terminal to the network.
- Identifying your terminal to the network.
- Selecting a host.
- Connecting to the selected host.
- Identifying yourself and your terminal to the host.
- Selecting a network application.

The first part of this appendix gives a brief description of the eight steps. The second part of the appendix gives a more detailed description of the same eight steps followed by additional general information. You may find the brief description of the steps sufficient to accomplish login at your site. If not, consult the detailed description. In addition to more detail, it provides shortcuts, the various options that are available at any given step, and failure conditions.

BRIEF DESCRIPTION OF ACCESS

STEP 1 - GATHERING INFORMATION

Find the answers to the following questions.

- Is your terminal hardwired to the system?
- Which key is the data transmission key for your terminal?
- If you have a dialup terminal, what telephone number must you dial?
- Into which of the following categories does your terminal fall?

  Asynchronous terminals
  Asynchronous terminals using X.25 protocol
  CDC mode 4 synchronous terminals
  Houston Automatic Spooling Program (HASP) terminals
  Bisynchronous terminals like IBM 2780/3780
  Bisynchronous terminals like IBM 3270

- Must you supply a family name, user name, password, and application name during login? If so, get the required information.
- Does your terminal use the ASCII character set and code?

STEP 2 - SETTING UP YOUR TERMINAL

- Turn power switch on.
- Load and initialize any software or controlware needed by your terminal.
- Set the terminal's duplex or echoplex switch to the correct position (typically HALF if your keyboard entries appear unduplicated on your console's screen or printer).
- Set the terminal's parity switch to its proper position.
- Set the line speed switch to its proper setting.
- Set the transmission mode to either the character or line mode position.
- Set the online/offline switch to the online position.

STEP 3 - CONNECTING YOUR TERMINAL TO THE NETWORK

Skip this step if your terminal is hardwired. Otherwise, determine which of the configurations in figures C-1, C-2, C-3, and C-4 most closely matches your terminal and follow the instructions in the figure.

1. Turn on coupler.
2. Pick up receiver.
3. Dial phone number.
4. Wait for high-pitched tone.
5. Fit receiver into coupler. Be sure you put the cord end of the receiver into the correct cup, as indicated on the coupler.

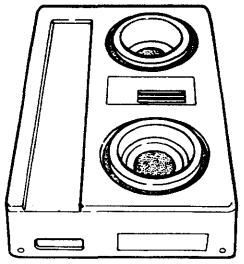


Figure C-1. Separate Accoustic Coupler

1. Pick up receiver.
2. Push button, such as ORIG button on Teletype model 33 or 35.
3. Dial phone number.
4. Wait for high-pitched tone.
5. Push button, such as ORIG button on Teletype model 33 or 35.
6. Replace receiver.

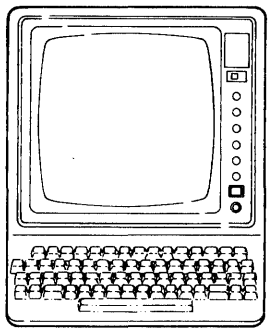


Figure C-2. Data Set Built into the Terminal

1. Pick up receiver.
2. Set switch to TALK or push TALK button.
3. Dial phone number.
4. Wait for high-pitched tone.
5. Set switch to DATA or push DATA button.
6. Replace receiver.

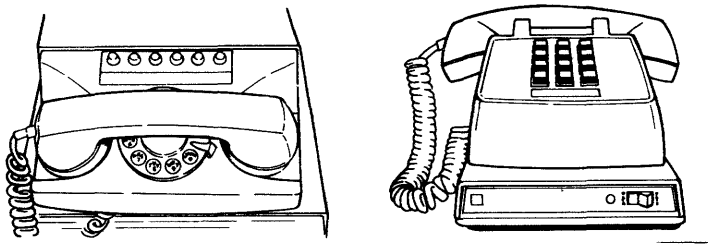


Figure C-3. Data Sets with Switches and Buttons

1. Pick up receiver.
2. Dial phone number.
3. Wait for high-pitched tone.
4. Fit receiver into coupler. Be sure you put the cord end of the receiver into the correct cup, as indicated on the coupler.

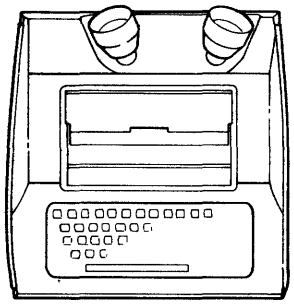


Figure C-4. Terminal with Built-in Acoustic Coupler

STEP 4 - IDENTIFYING YOUR TERMINAL TO THE NETWORK

If the network does not prompt you within a few seconds after you are connected to it (there might be a light on the terminal that indicates when your terminal is connected), perform the appropriate following steps.

- For asynchronous terminals not using X.25 protocol:

  a. Press the key that transmits a carriage return. The network responds with two line feeds when it recognizes your line speed.

  b. Type a closing parenthesis (not required if the terminal uses the ASCII character set and code) and press the carriage return key. When the network recognizes your character code set, it responds with a line feed.

- For asynchronous terminals using X.25 protocol:

  Get the information from your site.

- For mode 4 synchronous terminals:

  Press the data transmission key.

- For HASP and 2780/3780 bisynchronous terminals:

  Send a /*CONFIG card image.

- For 3270 bisynchronous terminals:

  No identification is necessary.

STEP 5 - SELECTING A HOST

If you have a choice of hosts or host paths (multihost), the network displays a list of the hosts and host paths that are available and prompts you to make a selection. Otherwise, you can enter:

```
ct HD
```

to get a list of hosts. ct is your terminal's network control character (refer to Volume 3, System Commands).

Make your selection with one of the following commands.

```
ct HN=node
      or
ct HS=hostname
```

node is the number of a host path; and hostname is the name of a host.

STEP 6 - CONNECTING TO THE SELECTED HOST

If your terminal was not automatically connected to a host, the network prompts you to connect. Enter a carriage return to make the connection.

STEP 7 - IDENTIFYING YOURSELF AND YOUR TERMINAL TO THE HOST (H2)

Once your terminal is connected to the host, the host displays a banner similar to:

```
WELCOME TO THE NOS SOFTWARE SYSTEM.
COPYRIGHT CONTROL DATA 1978, 198x.

date  time              terminal name
site identification  NOS 2
```

Following this banner, you may be prompted for a family name, then a user name, then a password, and finally an application name (if there is no default). For each prompt, enter the appropriate information followed by a carriage return.

STEP 8 - SELECTING A NETWORK APPLICATION

If your login is successful, your terminal is placed under the control of the requested or default application, which then prompts you for a command.

DETAILED DESCRIPTION OF ACCESS

The following description covers in more detail the eight steps briefly described earlier. Although each of the steps is always performed when a terminal accesses the system, you might find that your network's administrators can make many of these steps unnecessary by having the software perform the steps for you.

STEP 1 - GATHERING INFORMATION

If you know the answers to the following questions, or can get the answers from people at your site, the text following each question will tell you which steps and corresponding subsections of this appendix you can skip. If you cannot answer the questions, read all of the subsections; each procedure description contains hints for answering the questions.

1. Is the terminal hardwired, or is it a dial-up terminal?

   A hardwired terminal is connected directly to the network through a device called a modulator/demodulator (modem). A modem can be either a separate box, or built into the terminal. A dial-up terminal temporarily connects to the network through a telephone line and either an acoustic coupler or a telephone data set (a coupler or data set does the same things as a modem).

   If you do not need to dial a telephone number to use the network, then your terminal is hardwired. (A nearby telephone does not necessarily mean the terminal is dial-up.)

2. Is your network configured for automatic recognition?

   Your site can ask the network software to determine certain things about your terminal when the terminal joins the network. This process is called automatic recognition. [Terminals using X.25 packet-switching networks (PSN) cannot use automatic recognition.]

   If your network is configured for automatic recognition, you must perform an extra step each time you connect your terminal to the network. If your terminal is not so configured, you may skip questions 3 and 4.

3. What protocol does your terminal use?

   Your terminal must fit into one of the following categories.

   - Asynchronous terminals that work like teletypewriters or IBM 2741 terminals.
   - Asynchronous using an X.25 PSN through a packet assembly/disassembly (PAD) service; these terminals work like teletypewriters.
   - CDC mode 4 synchronous; mode 4A terminals work like the CDC 200 User Terminal, while mode 4C terminals work like CDC 711 or 714 terminals.
   - HASP terminals.
   - Bisynchronous terminals that work like IBM 2780 or IBM 3780 terminals.
   - Bisynchronous terminals that work like IBM 3270 terminals.

   If your terminal operating manual does not tell you what protocol your terminal uses, question 4 and the terminal classes shown in Volume 3, System Commands might help you answer this question. If your network is not configured for automatic recognition, skip question 4.

4. What class of terminal is it?

Many different kinds of terminals exist. The network software assigns terminals it supports to classes. If you know what the terminal class is, that will tell you the protocol, as follows:

Class	Protocol
1 thru 8	Asynchronous or X.25 asynchronous
9 and 14	HASP bisynchronous
10 and 15	CDC mode 4A synchronous
11 thru 13	CDC mode 4C synchronous
16 and 17	IBM 2780 and 3780 bisynchronous
18	IBM 3270 bisynchronous
Other	Site-defined

Terminals in classes 1 through 8 sometimes are called interactive terminals. A console has an input and output device that can be used for dialog with the network. Most consoles use a television-like screen or a typewriter-like mechanism to convey output information. Almost all consoles have a keyboard for entering information to the network. Some consoles might also use mechanisms such as magnetic tape cassettes or paper tape reader/punches for input or output.

Terminals in classes 9 through 17 sometimes are called remote batch terminals or remote job entry terminals. Such terminals have an interactive console and can also have batch devices the network can use for input only or for output only.

Figure C-5 shows a console with a display screen and a keyboard. Figure C-6 shows a console with a printer and a keyboard. Figure C-7 shows a remote batch terminal with a console, card reader, and a batch printer device.

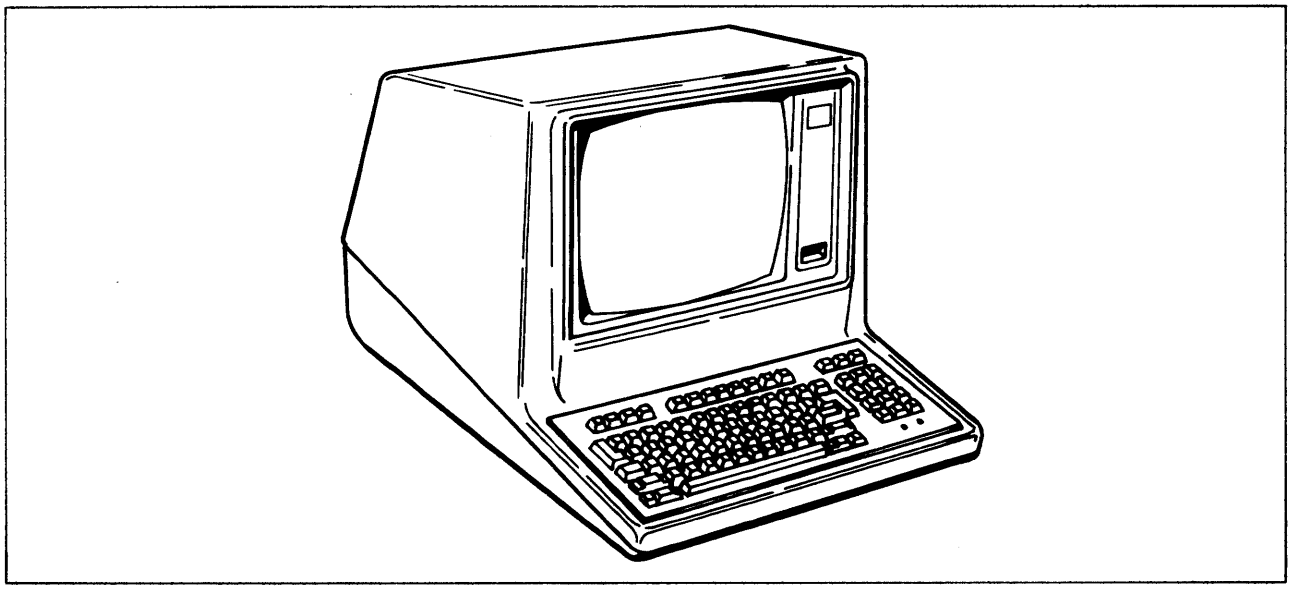


Figure C-5. Interactive Display Console

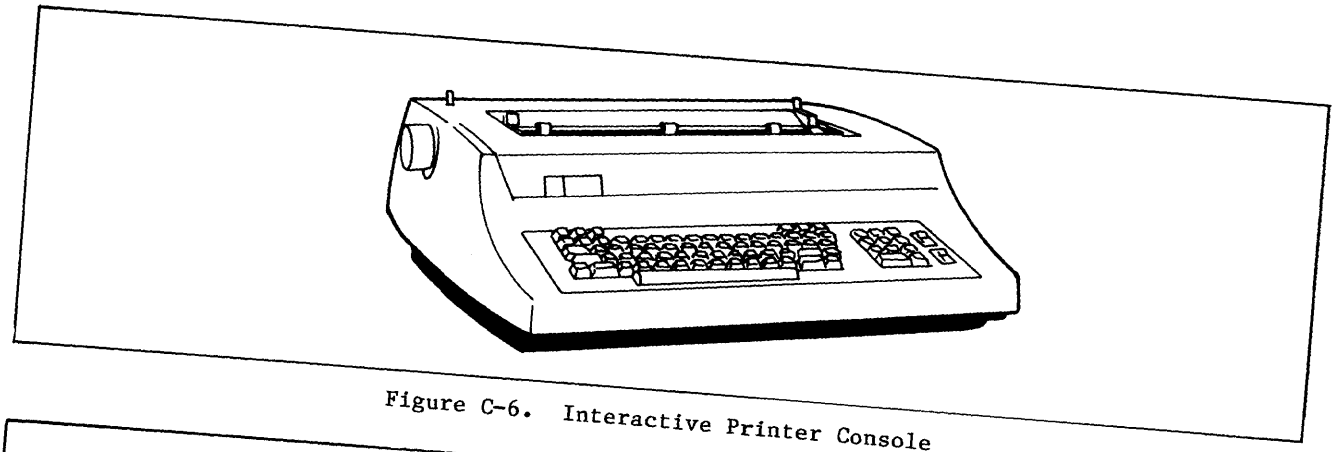


Figure C-6. Interactive Printer Console

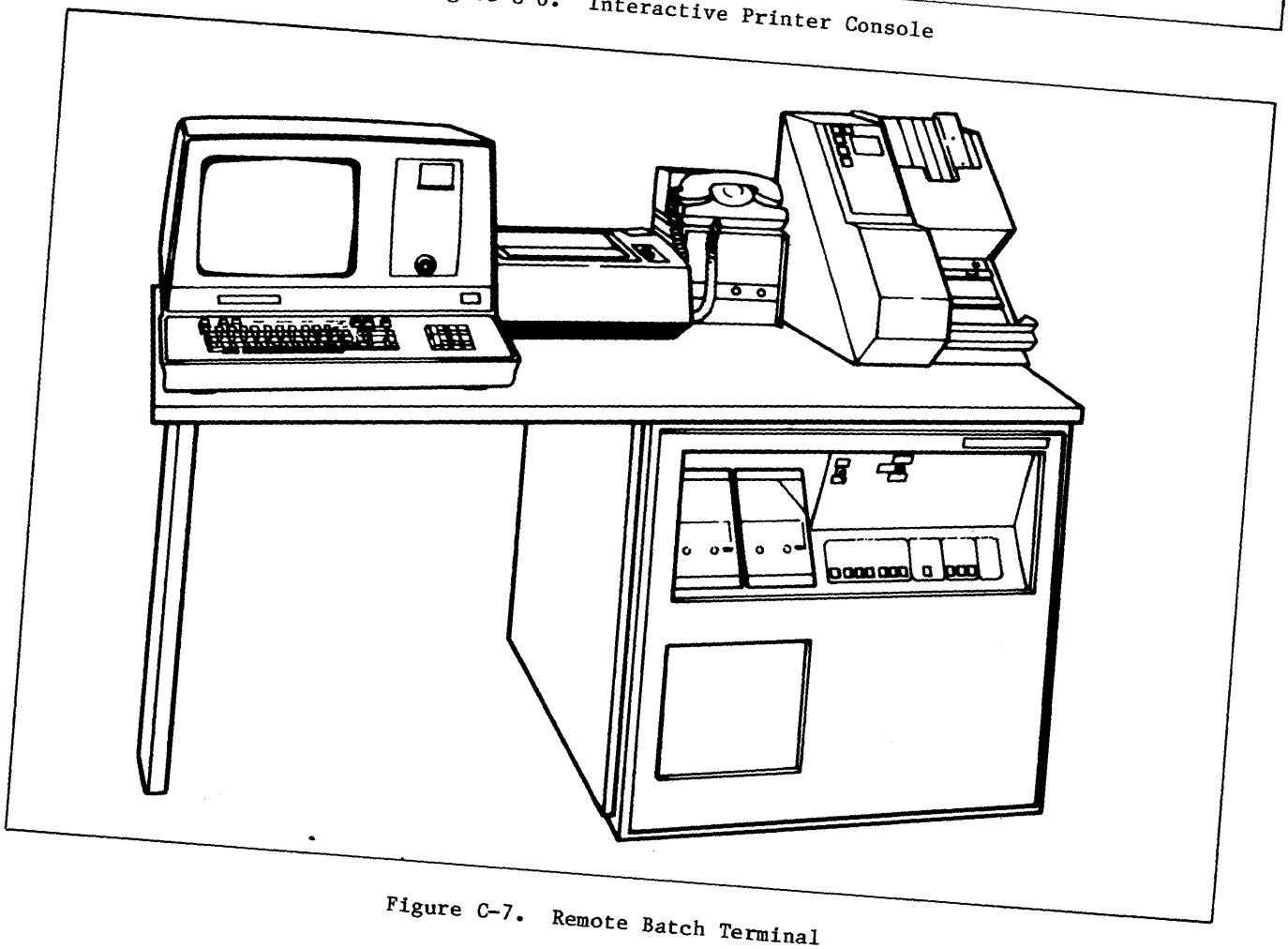


Figure C-7. Remote Batch Terminal

5. What key ends a line of data? What key sends data? (Does the terminal use block mode?)

   You should know when data leaves your terminal so that you know when to expect a response from the network or the host computer. Your terminal operator's guide should tell you what key or keys send data from your terminal.

   Terminals transmit information in one of the following ways.

   - A character at a time (transmission occurs as you enter each character).
   - A line at a time (transmission might occur when you end each line).
   - Several lines at a time, called a block (transmission occurs when you direct the terminal to transmit).

   Terminals that send data a character at a time are treated by the network as if they sent characters a line at a time. Until you end the line by entering a character or set of characters called an end-of-line character, you can backspace within the line and change data. When you enter the end-of-line character, the network processes the data.

   The end-of-line character used can be changed by you, by a site administrator, or by an application. We represent the end-of-line character for such terminals by a carriage return.

   This appendix assumes you enter an end-of-line character after each command.

   Terminals that send data a character at a time usually send the end-of-line character when you use a key called the end-of-line key (remember, the character can be changed). The end-of-line key might be labeled CARRIAGE RETURN, RETURN, RET, NEWLINE, NEXT, ATTN, or CR.

   Terminals that transmit a line at a time usually send the end-of-line character and transmit the line when you use the end-of-line key. If the character is changed, then you must enter the character as the last entry on the line before pressing the end-of-line key. The end-of-line key might be labeled CARRIAGE RETURN, RETURN, RET, NEWLINE, NEXT, ATTN, or CR.

   Terminals that send data a block at a time do not transmit your data to the network when you press an end-of-line key. Until you press a key called the end-of-block key, you can position the cursor anywhere within any unsent line and change data. This key might be labeled SEND, ETX, EOT, or XMIT. When you press the end-of-block key, the network processes each line of data as if it had been transmitted one line at a time.

   The network expects each block from a block mode terminal to end with a character or set of characters called the end-of-block indicator. The end-of-block indicator used can be changed by you, by a site administrator, or by an application program. We do not show the end-of-block indicator in the examples.

   Terminals that transmit a block at a time usually send the end-of-block character and transmit all lines in the block when you use the end-of-block key. If the character is changed, then you must enter the character at the end of the last line before pressing the end-of-block key.

   Terminals that transmit blocks are sometimes said to work in batch mode; do not confuse this kind of batch mode operation with remote batch data transmissions. Remote batch data transmissions are simply transmissions of files from or to the host using batch devices, bypassing the terminal.

6. What telephone number is appropriate?

   If your terminal is hardwired, you may skip questions 6 and 7.

   Your network might use a specific telephone number for terminals that communicate at a certain speed, for terminals operating in block mode, or for all terminals using certain protocols.

   The telephone number you use determines whether you need to complete some of the procedures. If you do not know what numbers you should use, get help from a site administrator.

7. Is your terminal automatically connected to a host?

   If it is, skip question 8.

8. What is the name or node number for the host you need to use?

9. Will your terminal be automatically logged in?

   If so, skip question 10.

10. What is the family name, user name, and password appropriate for you on the host system you will be using?

    You can get this information from site administrators.

11. Is your terminal automatically connected to a network application?

    If so, you will not have to select an application at login.

STEP 2 - SETTING UP YOUR TERMINAL

If your terminal is hardwired, you can probably skip most of this subsection; hardwired terminals are usually set up once and left in the appropriate condition. Before using this subsection, review the applicable terminal operator's manual, which describes the procedures for setting up your terminal. If you are using an X.25 terminal, follow the PAD access procedures supplied by the packet-switching network service.

Switch settings are important. The number, type, settings, and names of switches vary from terminal to terminal. Some terminals use buttons or toggles for switches; others use software option selections as switches.

If other people have used your terminal to connect to the network, the switches should be set correctly; do not change them. Directions for setting the switches may be posted near your terminal.

Here is a suggested procedure to follow when setting up your terminal:

1. Turn on the power switch.

2. Load and initialize any software or controlware needed by your terminal. You will find detailed information on this task in your terminal's operator manual.

3. Set the terminal's duplex or echoplex switch to the correct position (use HALF if you are not sure). If you wish to have the display of your login password suppressed, you must use full duplex. If nothing you enter appears on your console after you connect to the network, the network must display characters for your console. Volume 3, System Commands describes how to enable and disable character echoing by the network (refer to the EP command).

4. Set the terminal's parity switch to the proper position. If garbled output appears at your console after you connect to the network, the network must use a different parity choice for your terminal. Volume 3, System Commands describes how to change parity use by the network (see the PA command).

5. Set the line speed switch to a speed that matches one associated with the telephone numbers you were given. If you do not know the line speed you should use, set the switch to any position for X.25, mode 4 synchronous, or bisynchronous terminals; set the switch to 30 characters per second (300 bits per second) for asynchronous terminals.

6. Set the transmission mode switch to either the character or line mode position (you can change it later if you have a block mode terminal). For example: on a Viking 721, select the CHAR option; on a CDC 751, set the switch to the CHAR position; on a CDC 200 User Terminal, set the switch to the LINE position.

7. Set the online/offline switch to the position that permits online communications. For example: on a Viking 721, select the ON option of the LINE setting; on a CDC 713, turn off the LOCAL indicator switch; on a Teletype, set the LINE/OFF/LOCAL switch to the LINE position; on a CDC 200 User Terminal console, set the ATTENDED/UNATTENDED switch to the ATTENDED position.

You are now ready to connect your terminal to the network.

STEP 3 - CONNECTING YOUR TERMINAL TO THE NETWORK

To begin communication, your terminal must be physically connected to the network. If your terminal is hardwired, skip the remainder of this subsection; this subsection applies only to dial-up terminals.

A dial-up terminal uses either an acoustic coupler (asynchronous terminal) or a data set (asynchronous, X.25, bisynchronous, or synchronous terminal) to link itself to the network.

An acoustic coupler can be built into the terminal. Figure C-4 shows a terminal with a built-in acoustic coupler and gives directions to connect the terminal to the network.

An acoustic coupler can also be separate from the terminal. Figure C-1 shows an acoustic coupler separate from a terminal and gives directions to connect the terminal to the network.

A data set can be built into the terminal. Figure C-2 shows a terminal with a data set built in and gives directions to connect the terminal to the network.

A stand-alone data set can have either a switch labeled TALK and DATA or individual buttons labeled TALK and DATA. Figure C-3 shows both a data set with switches and a data set with buttons and gives directions to connect the terminal to the network.

After you dial the phone number, either you get a high-pitch tone or an operator answers. If an operator answers, ask for your terminal to be connected, wait for the tone, and then place the receiver in the coupler or set the data set switch or button to the on or data position. If you get a busy signal, wait and redial or try another number.

If your terminal has an indicator (sometimes marked DSR, DATA SET READY, or SYSTEM ACTIVE), it lights to let you know that the terminal is connected to the network. Wait approximately 2 seconds after the light comes on before entering data.

You are now ready for the next step. You might need to do one or all of the following:

- Identify your terminal to an X.25 network (this appendix does not describe this procedure).
- Identify your terminal to the CDC network.
- Select a host.
- Connect to the selected host.
- Identify yourself and your terminal to the host (log in).
- Select a network application.

The software usually sends you a message that helps you decide which step comes next. If nothing appears at your terminal within a few seconds, you probably need to identify the terminal to the network.

STEP 4 - IDENTIFYING YOUR TERMINAL TO THE NETWORK

Once you have established physical connection to the network hardware, you might need to identify the terminal to the network software.

This identification procedure, called automatic recognition, is not always needed. It does not apply to X.25 terminals. It does not apply if the communication line used by your terminal is not configured for automatic recognition.

If your terminal does not display any information within a few seconds of physical connection, you should complete one of the identification procedures described in the following subsections. The procedure you use depends on the terminal protocol, which you can determine if you know the terminal's type and/or class.

If you do not complete the identification procedure within the allowed time, the network disconnects a dial-up terminal. Hang up the phone and redial the number if you are using a dial-up terminal. You must restart the procedure for a hardwired terminal.

Procedure for Asynchronous Terminals

For an asynchronous terminal, which belongs in terminal classes 1 through 8, complete this procedure within 1 minute:

1. Wait about 2 seconds after the light that indicates your terminal is connected to the network comes on.

2. Press the carriage return key to identify the line speed used by your terminal. The network software responds with two line feeds.

3. If your terminal uses an APL, EBCD, or correspondence code character set, type a closing parenthesis; if your terminal uses an ASCII character set, you need not type in anything.

4. Press the carriage return key to identify the character and code set used by the terminal. The network software responds with a line feed.

The software sends you a message that helps you decide which step comes next. Turn to the subsection appropriate for the next step you must perform.

Procedure for HASP Terminals

HASP terminals belong either to terminal class 9 or 14. For automatic recognition of your HASP terminal, you might need to modify the signon block or enter a special statement.

The first information transmitted by a HASP terminal after it is connected to the network is called a signon block. A terminal might handle signon block transmission in one of several ways.

- Some terminals automatically transmit a signon block that begins with the characters:

  /*SIGNON

  These eight characters and the information following them imitate the card required by some IBM host systems for terminal identification. This transmission can occur without your knowledge or intervention if the information is built into the firmware or hardware.

- Some terminals require you to enter information for the signon block before they will transmit anything to the network; however, they only allow the block to contain information beginning with the eight characters /*SIGNON. These characters and other information might be required as:

  - A command from the console.
  - A card read from any card reader.
  - A card read from a specific card reader.

- Some terminals require you to enter signon block information before they will transmit anything to the network, but the content of the signon block is left up to you. Whether the information can be entered from the console or from a card reader depends on the terminal.

The network software cannot use information from a /*SIGNON card image. When such a card image is received in a signon block, the signon block contents are discarded.

Instead, the network software uses a /*CONFIG statement for the automatic recognition procedure. You can send the /*CONFIG statement in the signon block when your terminal permits you to provide the contents of that block.

If the signon block does not contain the /*CONFIG statement and the terminal must use automatic recognition, a prompt requesting that statement is sent to the terminal console. When a prompt for the statement appears, enter the statement through a card reader.

The format of the /*CONFIG statement is:

```
/*CONFIG,ti,CO=co,CR=x,LP=x,CP=y,PL=z
```

Blanks are not allowed within the statement; a blank ends parameter processing. The parameters, which you can specify in any order, are:

Parameter	Description
ti	Terminal type: **POST** HASP postprint (default); this is terminal class 9. **PRE** HASP preprint; this is terminal class 14.
CO=co	Configuration ordinal, a decimal integer from 1 to 255; the default is 1. Use the configuration ordinal to select one of several terminal definitions defined for the line that describes a specific combination of additional characteristics. Site administration personnel can tell you the correct number to use.
CR=x	Identifies available card readers. The default is 1.
LP=x	Identifies available line printers. The default is 1.
CP=y	Identifies available card punches; y cannot equal z. The default is 1.
PL=z	Identifies available plotters; y cannot equal z. No default exists for z; you cannot omit PL=z if you have a plotter.

x, y, or z is a list of numbers specified by either:

```
1/2/ . . . /7
```

or:

ALL (all numbers from 1 through 7)

The numbers you use for x, y, and z must match the HASP stream numbers used for the corresponding devices within the workstation. Site administration personnel should have this information if you do not know what numbers to use.

For example:

```
/*CONFIG,PRE,CO=2,CR=2,LP=2/3,CP=4,PL=5
```

identifies your terminal as the second preprinting workstation defined for the communication line, having:

- A card reader and line printer on stream 2.
- Another printer on stream 3.
- A card punch on stream 4.
- A plotter on stream 5.

If no prompt for a /*CONFIG statement appears, you are ready for the next step.

The software sends you a message that helps you decide which step comes next. Turn to the subsection appropriate for the next step you must perform.

Procedure for Mode 4 Terminals

Mode 4 terminals belong to terminal classes 10 through 13 and 15. For automatic recognition, press the end-of-block key within 1 minute of physical connection.

You are ready for the next step. The software sends you a message that helps you decide which step comes next. Turn to the subsection appropriate for the next step you must perform.

Procedure for Bisynchronous Terminals

Bisynchronous terminals belong to terminal classes 16 and 17. Enter a /*CONFIG statement to specify the type of terminal and the devices available at it.

The format of the /*CONFIG statement is:

```
/*CONFIG,ti,CO=co,CR,LP,CP=y
```

Blanks are not allowed within the statement; a blank ends parameter processing. The parameters, which you can specify in any order, are as follows:

Parameter	Description
ti	Terminal type:
	2780 IBM 2780 (default); this is terminal class 16.
	3780 IBM 3780; this is terminal class 17.
CO=co	The configuration ordinal, a decimal integer from 1 through 255; the default is 1. Use the configuation ordinal to select one of several terminal definitions defined for the line that describes a specific combination of additional characteristics. Site personnel can tell you the correct number to use.
CR	Indicates an available card reader; assumed if not specified.
LP	Indicates an available line printer; assumed if not specified.
CP=y	Indicates an available card punch; assumed not to exist if not specified. The =y portion is not allowed for 2780; for 3780, the y portion is required and can be either 2 or 3 to correspond to the device selection character DC2 or DC3.

For example:

```
/*CONFIG,3780,CO=4,CR,LP,CP=2
```

identifes your terminal as a 3780 terminal having:

- A card reader and line printer.
- A card punch, which is selected by the character code DC2.

If no prompt for a /*CONFIG statement appears, you are ready for the next step.

The software expecting the next step sends you a message which helps you decide which procedure comes next. Turn to the subsection appropriate for the next step you must perform.

STEP 5 - SELECTING A HOST

You might need to select the host, which runs the software you want to use. This is not necessary if your installation personnel select a host path for your terminal when the network is configured.

You can select or change a site-defined host path after you connect your terminal to the network. The host path selected is used until either you change it or until your terminal is disconnected from the network.

More than one host path can be available to each host from a terminal. If one host path fails, you can select an alternate route to the host.

Selecting a Host

You can select a host or override a site-defined host selection with the following terminal definition command.

```
ct HS=hostname
```

ct represents the network control character for your terminal. For most asynchronous terminals this is ESC or %.

Each host may be configured with a unique name. This name (1 through 7 characters) is indicated by hostname.

For example:

```
%HS=ARHNOS
```

would select the path with the least traffic to the host called ARHNOS (% represents your terminal's network control character).

If you specify the name, your terminal uses the host path with the least traffic.

If you do not know the name you need, the host availability display (HAD) can show you what names are available. That display is described later in this appendix.

Once you have selected your host, you are ready for the next step.

Selecting a Specific Host Path

Use the following terminal definition command to select your own host path or to override the site-defined host path.

```
ct HN=nn
```

ct represents the network control character for your terminal. For most asynchronous terminals this is ESC or %.

Each access has a unique number, called a node number. The host node number (1 through 31) is indicated by nn.

For example:

```
%HN=2
```

would select access 2 (% represents your terminal's network control character).

If you omit the node number, your terminal uses the host path the network is using to communicate with its supervising host.

If you do not know the node number you need, the host availability display (HAD) can show you what numbers are available. That display is described in the following subsection. Each node number may have a name for its host, provided by your site to help you identify the corresponding host system on the HAD.

Once you have selected the host path, you are ready for the next step.

The software sends you a message that helps you decide which step comes next. Turn to the subsection appropriate for the next step you must perform.

Controlling the Display of Host Paths

The host availability display (HAD) (figure C-8) lists all host paths your terminal can currently use to gain access to the host and all hosts in the network. You can turn display of the HAD on and off or request the display with the following terminal definition command.

```
ct HD=option
```

ct represents the network control character for your terminal. For most asynchronous terminals this is ESC or %.

If the display mode is N (no), you receive only two lines at your terminal (the host status message and the prompt message) instead of the full display when you disconnect your terminal from a host.

If you enter HD=Y or HD, the full display is issued immediately and when you disconnect your terminal from the host. You then have a choice of hosts and/or host paths to which you can request connection.

If you have a multihost system, typically the HAD appears when you first access the network. This appearance is controlled by site installation.

Display

```
host status message                    CONTROL CHARACTER = ct
NPU NODE= xx TERMINAL NAME= yyyyyy
HOST          NODE                     SELECTED/          STATUS
                                       CONNECTED

host1         node1                    condition1         status1
host2         node2                    condition2         status2
  .             .                        .                  .
  .             .                        .                  .
  .             .                        .                  .
hostn         noden                    conditionn         statusn

prompt message
```

Field	Description
host status message	Status of a given host, which can be one of the following:

Message	Meaning	You Also See:
HOST CONNECTED	You are connected to the host.	prompt message d or e
NO HOST SELECTED	Host is available; you have not selected a host.	prompt message b

Figure C-8. Host Availability Display (Sheet 1 of 2)

Field	Description		
	Message	Meaning	You Also See:
	NO HOST AVAILABLE	No hosts are available.	prompt message a
	HOST UNAVAILABLE	You have selected a host that is not up.	prompt message a
	HOST BUSY	Connection rejected by host.	prompt message a
	HOST DISCONNECTED	Connection to host terminated.	prompt message c or e
	HOST AVAILABLE	You have selected host that is up.	prompt message c
	NO HOST CONNECTED	You are not connected to a host.	prompt message c

ct — Network control character currently defined for your terminal.

xx — The number of the host node you are connected to.

yyyyyy — The terminal number of your terminal.

$host_i$ — The 1- to 7-character name of a host; if the network only has one host, this can be blank.

$node_i$ — The host node number used in selecting a path through the network; $1 \leq node_i \leq 31$.

$condition_i$ —
S = selected, not connected
C = connected, not selected
SC = selected and connected
SA = selected, attempting connection

$status_i$ — Either AVAILABLE or NOT AVAILABLE; a host is only available when it is connected to the network.

prompt message — Action to take in response to a host status message, which can be one of the following:

prompt message a: ENTER ct HD TO SEE HOST STATUS
prompt message b: ENTER ct HN=nn TO SELECT HOST
prompt message c: ENTER INPUT TO CONNECT TO HOST
prompt message d: READY FOR INPUT
prompt message e: TERMINAL DISABLED BY NOP
prompt message f: INPUT DISCARDED

Figure C-8. Host Availability Display (Sheet 2 of 2)

STEP 6 - CONNECTING TO THE SELECTED HOST

Your site personnel might have configured your terminal so that it is automatically connected with a host. If so, the network continues to attempt connection indefinitely either until the connection is completed or until you stop it by entering ct TM. You can tell the network is attempting to connect you when the HAD shows a SA status for any host. If your terminal is automatically connected to a host, you may skip this subsection. If not, you will have to ask for the connection to be made.

You can ask to be connected to the selected host by entering a carriage return. If the carriage return is not preceded by an empty line, the information is discarded (unless it is a valid terminal definition command).

If your connection is rejected, you are notified. If you select another host path you must make another entry to complete a connection. The network software physically disconnects dial-up devices if you do not attempt connection to a host within 2 minutes of completing the previous step.

Once you are connected to a host, you are ready for the next step.

STEP 7 - IDENTIFYING YOURSELF AND YOUR TERMINAL TO THE HOST

The step that identifies you and your terminal to the host system is called login. Your terminal is identified to the host system by assigning it a family name, a user name, and a password. The host can prompt you for each of these things.

Your site can set up your terminal connection so that part or all of your login is done automatically. If so, you might not see the corresponding prompt and you need not enter the information.

Your site might also require use of a secure login initiation sequence, prior to login, to ensure that you send your user identification to the correct host software. If required, the sequence contains a site-selected security character and the sequence will be recognized by the system at any time. Depending on your terminal type, the secure login initiation sequence is entered in one of the following ways.

- At an asynchronous terminal (not 2741), press the interactive interrupt key (BREAK or ATTN) and type the character at a 2741 terminal, press the NL key twice. Then enter the ATTN key, the character, and press the NL key.
- At an X.25 terminal, press the interactive interrupt key, type the character, and press the message transmission key.
- At a mode 4 terminal, clear the screen, press SEND, clear the screen again, type the character, and press the end-of-block key.
- At a HASP terminal, type the character and press the end-of-line key.
- At a 2780 or 3780, enter /* and the character from the console or a card reader.

This action momentarily disconnects you from the host. You might need to reconnect your terminal to the host (if connection is not automatic). Now you are guaranteed that you are sending your login entries to the correct host software.

Refer to figure C-9 for a sample login from an asynchronous terminal. In the example, the terminal has not been configured by site personnel for automatic login. Everything you would typically enter is shown in lowercase.

```
WELCOME TO THE NOS SOFTWARE SYSTEM.
COPYRIGHT CONTROL DATA 1978, 198x.

84/04/14. 09/51.06. T05A50
CDC NETWORK OPERATING SYSTEM  NOS 2
FAMILY:
USER NAME: jrc3069
PASSWORD:  janc
/T05A50  - APPLICATION: iaf
```

Figure C-9. Sample Login from an Asynchronous Terminal

Automatic Login

The family name or user name assigned to your terminal for an automatic login can be:

- A mandatory value.
- A default value.
- A primary value.

If a mandatory value is assigned, you do not receive a prompt and cannot use any other value.

If a default value is assigned, you are prompted and you can either: enter an empty line as a response to the system prompt if you want either the default family or user name to apply, or respond to the prompt with another value.

The default family name value assigned to your terminal for automatic login can be different from the default value used by the host for a family name. You can override the login default even when you do not know the system default; this procedure is described in the following subsection.

If a primary value is assigned, you do not receive a prompt the first time login occurs while you are connected to the host. Subsequent logins (refer to Restarting Login Identification) do prompt you. You can override a primary value for user name by entering a different value during an abbreviated login. Abbreviated login is described in a later subsection.

Login Dialog

Login begins when the system displays the following lines.

```
WELCOME TO THE NOS SOFTWARE SYSTEM.
COPYRIGHT CONTROL DATA 1978, 198x.
```

Then a line appears with the date, time, and terminal name; for example:

```
83/12/28. 15.23.58. TERM201
```

Next is a line your site supplies that identifies the host system used; something similar to:

```
CDC NETWORK OPERATING SYSTEM  NOS 2
```

The next line is either of the following prompts.

FAMILY:

This indicates that no mandatory or primary family name has been assigned to your terminal. Enter the 1- to 7-character name of the storage device that contains your permanent files and press the carriage return. If you want to use the default login family name, just press the carriage return after the FAMILY prompt appears. You can also enter a value of 0 to override a preassigned login default family name and use the system default family name. If you have no preassigned default family name, carriage return selects the system default family name.

USER NAME:

This indicates that no mandatory or primary user name has been assigned to your terminal. Enter the user name you were given. The user name, which can contain any combination of digits, letters, and asterisks, identifies you as the terminal operator.

The next prompt appears only after a USER NAME prompt has appeared. This final prompt is either:

```
PASSWORD:
```

or

```
PASSWORD:
XXXXXXX
```

The system attempts to preserve your password's secrecy. If you have set your terminal at full duplex and disabled echoplex (refer to Volume 3, System Commands), your password will not be displayed when you enter it. You will not see what you type in. At some terminals, the network overprints several characters on a line and asks you to type your password on that line, as shown by the second PASSWORD prompt. After creating this row of overstruck characters, the cursor moves back to the first character.

Enter the password currently associated with your user name. If you must enter the user name, you also must enter the password. No default exists for a password.

You must respond to each prompt within 2 minutes. If you take too long to respond, this message is displayed at your terminal.

```
TIMEOUT.
```

Your terminal is then disconnected from the host.

If you enter an unacceptable family name, user name, or password, you receive either the following message:

```
IMPROPER LOGIN, TRY AGAIN.
FAMILY:
```

if no mandatory family name is assigned to your terminal, or:

```
IMPROPER LOGIN, TRY AGAIN.
USER NAME:
```

if a mandatory family name is assigned. Reenter all needed login information, beginning with the requested parameter, regardless of when the error occurred.

You must spell each login entry correctly. If you make a mistake, such as typing the letter O for the numeral 0 or the number 1 for the letter l, your login will not be successful. Try again when the prompting sequence restarts.

You are allowed four consecutive chances to complete the login procedure. If you make four unsuccessful attempts to complete login, this message appears at your terminal:

```
USER RETRY LIMIT.
```

Your terminal is then disconnected from the host. If this happens, check the spelling of all of the entries. If you spelled everything correctly, contact a site administrator.

When the family name, user name, and password are accepted by the host, you must next connect your terminal to an application, if there is no default network application.

STEP 8 - SELECTING A NETWORK APPLICATION

Your site may be set up so that your terminal is connected to the IAF application automatically. Or your site may connect your terminal automatically to another application, depending on the installation. If so, you might not see the corresponding prompt. If not, you must complete this step yourself.

You are allowed several chances to make a correct entry. However, if you fail to connect successfully after four consecutive attempts, your terminal is automatically disconnected from the host. If this happens, check the spelling of the name you used. If the spelling was correct, contact a site administrator.

Automatic Connection

Your host may permit you access only to one network application, or an application can be connected automatically. Your host site can configure your connection for either possibility by preassigning an automatic connection application name.

A preassigned automatic application connection can be either mandatory or primary for the connection or it can be mandatory for the user. If either you or your connection has a preassigned mandatory application, you are not prompted, you are automatically connected to that application, and you cannot use any other application. If a mandatory application program is not running in the host, you are logged out and disconnected from that host.

If a primary value is assigned, you are not prompted for the initial connection request after identifying your terminal to the host, and connection to that program occurs automatically. If that connection is not successful or if you disconnect from that application program, you have a choice of actions; you can select another application or disconnect from the host.

To override a primary automatic application connection on the first attempt to select an application, name a different application in an abbreviated login entry (described later in this appendix) following a login prompt.

When a primary value is assigned, you are prompted on subsequent attempts to select an application, such as when you are switching applications. When you are prompted to enter an application name and you want to be connected to your primary application, enter an empty line as a response.

Manual Connection

This procedure begins when you see the following prompting message.

```
terminalname - APPLICATION:
```

The terminalname variable on this line is the same terminal name as the one on the first line of your login sequence. If your terminal is automatically logged into a network application, you do not receive this prompt.

When you receive the APPLICATION prompt, enter the letters and digits that identify the network application you want to access; for example, IAF for the Interactive Facility. You can access CDC-written applications from the following list that are installed at your site:

- Remote Batch Facility (enter RBF).
- Interactive Facility (enter IAF).
- Transaction Facility (enter TAF).
- Terminal Verification Facility (enter TVF).
- Message Control System (enter MCS).
- PLATO (enter PNI, which stands for the PLATO-NAM Interface).

Other applications, written at your site, might be available.

You must respond to the prompt within 2 minutes. If you take too long to respond, this message is displayed at your terminal.

```
TIMEOUT.
```

Your terminal is then disconnected from the host.

You must spell your entry correctly. If you make a mistake, such as typing the letter O for the numeral 0 or the letter l for the number 1, you receive the following error message:

```
ILLEGAL APPLICATION, TRY AGAIN.
terminalname - APPLICATION:
```

The terminal name is indicated by terminalname. You can also receive this message if your user name is not permitted access to the application you named. If you did not spell the name correctly, try again. If you did spell the name correctly, disconnect your terminal and contact a site administrator.

If you make four unsuccessful attempts to enter an application name, this message appears at your terminal.

```
APPLICATION RETRY LIMIT.
```

Your terminal is then disconnected from the host.

This step is complete when the application you request connects to your terminal. The application you asked to use might print an identification line next.

Whether the program sends such a message or not, you can use commands recognized by that application. If you need to disconnect your terminal from the application, the application probably has a command for that purpose. If the program has no such command, the subsection called Disconnecting from a Host describes a command you can use.

ABBREVIATED LOGIN AND APPLICATION CONNECTION

You can shorten the standard login and application connection procedures by entering all of the required information at once. For example, in response to the FAMILY prompt, you can enter the following information on the same line.

```
FAMILY: familyname,username,password,application
```

Separate all entries with a comma. The entries are order-dependent. Use a comma to indicate any default or unused entries.

You do not receive any of the remaining login selection prompts if you use this abbreviated login procedure, unless you omit a required entry or supply an entry that is not recognized. Your password entry is not protected by the system when you use an abbreviated login procedure.

Here are some examples. To log in and connect to the Interactive Facility application IAF with a family name of SYSTEMA, a user name of RLS4525, and a password of PASS123, enter:

```
SYSTEMA,RLS4525,PASS123,IAF
```

To use the site-defined login default family name, enter:

```
,RLS4525,PASS123,IAF
```

To override your site-defined login default family name with the host's default family name, enter:

```
0,RLS4525,PASS123,IAF
```

To use a login default family name and user name defined by the site for your terminal, enter:

```
,,,IAF
```

To also use a default application name defined by the site for your terminal, enter:

```
,,,
```

You can supply combined entries in response to the FAMILY, USER NAME, and PASSWORD prompts. Although you do not need to complete the remaining sequence on the same line, you must enter the information in the correct order: family name, user name, password, and application (if there is no default). For example:

```
FAMILY: systema
USER NAME: rls4525,pass123
terminalname - APPLICATION: iaf
```

If you supply a user/password combination that is overridden by a mandatory specification, this message is displayed.

```
VOLUNTEERED USER/PASSWORD IGNORED.
```

If you offer an application name that is overridden by a preassigned mandatory application, the following message is displayed.

```
VOLUNTEERED APPLICATION IGNORED.
```

SWITCHING TO A DIFFERENT APPLICATION

Unless you have a mandatory network application assigned for your terminal, you can transfer your terminal's connection from the current application to another application. There are two ways to do this, depending on the commands used by the current application.

Direct Switches

You can switch in one step by entering a switch command recognized by the current application. The following switch commands are recognized by CDC-written network applications:

Application	Command
RBF	IAF
IAF	BYE,application
MCS	LOGIN,application

For example, if your terminal is connected to RBF, you can gain access to IAF by entering the IAF command.

Do not enter any more input until the application acknowledges it has processed your switch command. Otherwise, you may lose the input.

Indirect Switches

If your current application does not have a command for a direct switch, you must disconnection your terminal from the current application with a command recognized by that application. Then you must request another application connection from the host. The following disconnection commands are recognized by CDC-written network applications:

Program	Command
RBF	END
IAF	BYE,
MCS	END
TAF	EX.LOGT
TVF	END

For example, you can switch your terminal connection from RBF to an application other than IAF by entering the END command. You then receive a prompt for a new application name. Do not enter any more input until the application acknowledges it has processed your disconnection command. Otherwise, you might lose the input.

Dialog for Direct and Indirect Switches

After disconnecting the application, the system indicates that a connection switch is in progress by issuing this message.

```
application - CONNECT TIME hh.mm.ss.
```

The application that was just disconnected is indicated by application and the time that passed while the terminal was connected to the program is indicated by hh.mm.ss.

If you used a switch command, you receive an identifying message from your new application. If you used a disconnection command, you receive another prompt for an application name.

```
terminalname - APPLICATION:
```

unless the terminal is permitted access to only one application. If so, you are disconnected from the host.

If you enter the name of another application in response to the APPLICATION prompt, you are connected to that application.

RESTARTING LOGIN IDENTIFICATION

There are many reasons why you might want to restart login identification and the subsequent application connection. Perhaps you have several user names available to you, each with permission to use different host resources. Or perhaps someone else left the terminal connected to an application you do not need, and you do not want to use their user name.

There are three ways to restart login, depending on the commands recognized by the current application and installation options.

Direct Restarts

You can restart host identification by entering a login command recognized by the current application. The following login commands are recognized by CDC-written network applications:

Application	Command
RBF	LOGIN
IAF	HELLO
MCS	LOGIN

For example, if your terminal is connected to IAF, you can restart login by entering the HELLO command.

Do not enter any more input until the application acknowledges it has processed your command. Otherwise, you may lose the input.

Indirect Restarts

If your current application does not have a command for a direct switch, you must disconnect your terminal from the current program with a command recognized by that application. Then you must request restart of login from the host. The disconnection commands recognized by CDC-written network application are described earlier in this section under Switching to a Different Application.

For example, you can restart your terminal login from RBF by entering the END command. You then receive a prompt for a new application name. Do not enter any more input until the application acknowledges it has processed your disconnection command. Otherwise, you might lose the input.

Secure Login Restarts

If your site has selected the secure login initiation option and defined a security character, the sequence may be entered at any time to immediately disconnect from the current application and initiate the user login procedure.

After entering the disconnection command, you receive another prompt for an application name:

```
terminalname - APPLICATION:
```

unless the terminal is permitted access to only one application program. If so, you are disconnected from the host and can restart login by reconnecting.

If you receive the APPLICATION prompt, you use one of the following commands to ask the host system to restart login:

```
HELLO

LOGIN
```

DISCONNECTING FROM A HOST

You might want to disconnect your terminal from a host for several reasons, for example:

- You want to disconnect from the network properly, without leaving behind an indication that something went wrong; disconnecting from a host is the first step in disconnecting from a CDC network.
- You are allowed access to more than one host system, and you want to transfer your terminal's connection from the current host to another host.

There are four ways to do this, depending on the commands used by the current application.

Disconnecting with Application Commands

You can disconnect in one step by entering a host disconnection command recognized by the current application. The following host disconnection commands are recognized by CDC-written network applications:

Program	Command
RBF	LOGOUT or LOGOFF
IAF	BYE or LOGOUT or GOODBYE
MCS	BYE or LOGOUT

For example, if your terminal is connected to IAF, you can disconnect from the host by entering the BYE command.

Disconnecting with Host Commands

If your current application does not have a command for direct disconnection, you must first disconnect your terminal from the current program with a command recognized by that application. Then you must request host disconnection from the host. Application disconnection commands are described earlier in this section under Switching to a Different Application.

After disconnecting the application, the system indicates that a connection switch is in progress by issuing this message:

```
application - CONNECT TIME hh.mm.ss.
```

The application that was just disconnected is indicated by application and the time that passed while the terminal was connected to the program is indicated by hh.mm.ss.

You then receive another prompt for an application program name:

```
terminalname - APPLICATION:
```

unless the terminal is permitted access to only one application. If so, you are disconnected from the host anyway.

The following host disconnection commands are valid responses to the APPLICATION prompt on CDC host systems:

```
BYE
```

```
LOGOUT
```

For example, you can disconnect your terminal from RBF by entering the END command. You then receive a prompt for a new application name and enter BYE.

Disconnecting with a Network Software Command

You normally end a connection with a host by using a command to disconnect you from an application and/or a command to log out of the host. Events might make it impossible for you to do either. When this happens, you can end your terminal's connection to the host by entering this terminal definition command.

```
ct TM carriage return
```

ct represents the network control character. For most asynchronous terminals this is ESC.

The network responds to your TM command with a new host availablity display. You can use the TM command if an application does not respond to your entries.

Reconnecting to a Host or Connecting to a Different Host

For a single host system, after your terminal is disconnected from the host, this message is printed.

```
HOST DISCONNECTED.  CONTROL CHARACTER = ct
```

ct is the network control character. This message is followed by the prompting message.

```
ENTER INPUT TO CONNECT TO HOST
```

You can then attempt another connection to the same host by sending any input you want. If you have a multihost system, typically the HAD will be displayed between these two messages. To reach a different host, you must complete the procedures for Selecting a Host and Connecting to the Selected Host described earlier in this appendix.

Disconnecting from One Host and Connecting to a Different Host

You can disconnect from one host establish a connection with another host using the single command

```
ct  HC = hn
```

ct is the network control character, and hn is the name of the desired host.

INTERRUPTIONS

Events can sometimes interrupt your input or output. These interruptions fall into two categories.

- Interruptions for communication with the network operator.
- Temporary suspensions in communication.

Communicating with the Network Operator

The term network operator includes any operator who has access to the network, including the host operator. Any network operator can send and receive messages that involve you.

Receiving Messages

The network operator can send short messages to a terminal. The message appears at your terminal in this format.

```
FROM NOP . . . message text
```

These messages can appear at any time a new line of output is possible at your terminal. For example, the network operator might want to inform all connected terminals that communication is going to be suspended intentionally with this message.

```
FROM NOP . . . SHUTDOWN IN 5 MIN
```

Sending Messages

You can respond to such messages by entering the following terminal definition command.

```
ct MS=message carriage return
```

ct represents the network control character. For most asynchronous terminals this is ESC.

The message text must not exceed 50 graphic characters. Because the network operator can be located at a terminal that does not support many special characters, you should avoid characters not in the ASCII 63-character subset described in appendix A.

You can also begin dialog with the network operator. For example, if you need to use a remote batch printer that has been logically disabled, the entry:

```
%MS=PLEASE ENABLE LP=M4C555
```

causes the message PLEASE ENABLE LP=M4C555 to appear at the network operator's terminal, prefixed by the terminal name of your terminal. The % character represents the network control character for your terminal.

Preventing Messages

Sometimes it is important that output to your terminal does not include unplanned messages from the network operator. You can prevent delivery of messages from the network operator by locking them out with the following terminal definition command.

```
ct LK=option
```

ct represents the network control character. For most asynchronous terminals this is ESC.

After you enter this command with the option Y, any message to your terminal from the network operator is discarded by the network software. Messages are discarded until you again allow delivery by using the command with the option N.

Site administrators can configure your terminal so that messages are discarded as soon as the terminal is connected to the network. A network application can also change the LK option in use.

Suspensions of Communication

Sometimes, message traffic in the network is so heavy that all storage is temporarily used up. Also, a host may be too busy to accept data. During these times, the network software takes steps that temporarily stop input until enough output has occurred to free additional storage.

Although storage may be low, sometimes the network cannot prevent you from entering data. Therefore, the network discards each message from your terminal and sends this message to your terminal.

```
WAIT . .
```

When the following message appears, you can reenter any information that was previously discarded.

```
REPEAT . .
```

If your terminal transmits more than one message at a time, you must determine which information you need to reenter. The application might have commands that can help you determine which messages it received.

Communication Failures

Events can occur that:

- Cause your application to fail.
- Disconnect your terminal from an application.
- Disconnect your terminal from the host.
- Disconnect your terminal from the network.

Application Failures

If an application fails after your terminal is connected, this message is displayed.

```
APPLICATION FAILED.
application CONNECT TIME hh.mm.ss.
terminalname - APPLICATION:
```

The length of time your terminal was connected to the application is specified by hh.mm.ss; application is the name of the application; and terminalname represents the terminal name the network uses to identify the terminal. You either can select another application or log out.

Disconnection from an Application and/or a Host

An application is informed of two events that you might want to take action on.

- Long periods of device inactivity.
- Pending network shutdown.

The network software informs an application if more than 10 minutes have elapsed without any communication between a device at your terminal and the application. An application can:

- Ignore this time interval.
- Ask you for continued dialog.
- Disconnect from your terminal or from the device with or without telling you.

The effect of device inactivity depends on the application with which the device is currently connected.

When shutdown of the network is pending, the network operator can enter a command that informs all applications of the pending event. Your application might take actions that either result in connection termination or application failure. It also might send you a message to warn you of the event.

If you receive a message from an application indicating that shutdown is pending, you should take steps immediately to save all data or files you are currently using. Then end connection with both the application and the host as soon as possible.

Communication Failures

The network informs you that it has lost communication with the host software by displaying the host status message:

```
HOST DISCONNECTED.  CONTROL CHARACTER = ct
```

followed by the prompting message:

```
ENTER INPUT TO CONNECT TO HOST
```

If you have a multihost system, the HAD will be displayed between these two messages.

If you are at a dial-up terminal and you do not attempt to connect to the same or a different host within 2 minutes, your terminal is disconnected from the network.

If communication between the network and the host software resumes, you must connect to the host and log in again but you do not need to access the network again (perform dial-up and auto-recognition procedures). These host communication failures occur either when the host has been shut down or when hardware fails in the communication path between the network and the host.

All network software conditions (that is, parameters that may be set with terminal definition commands) are retained across host disconnection except for transparent, full ASCII, and special edit input modes. Refer to Volume 3, System Commands for a discussion of these commands.

Failures in the communication path between the network hardware and your terminal are treated by the network software as if your terminal had failed. These failures can occur because of hardware problems, or maybe because you entered an input character during output. If you hold the interactive output interrupt key down for too long, a communication failure can occur. This key is identified in Volume 3, System Commands.

There are several tests for communication failures. If your terminal has a CTS, ON-LINE, or CARRIER indicator, the light goes out when a failure occurs. At other terminals, if a failure has occurred, requesting the host availability display produces no response from the network.

Batch input and ouput devices associated with the failed console stop running, and batch device connections are ended. The network software does not distinguish between dial-up and hardwired terminals when processing terminal failures.

Terminal failures require you to complete new access and login procedures to resume communication with the network. Terminal failures always cause disconnection when the terminal is on a dial-up line.

DEBUGGING AIDS D

Some program errors prevent compilation or assembly of the source program; other errors prevent execution of the object program. You determine the cause of a compilation error using the compiler diagnostics, a source listing, and the compiler reference manual. The cause of an execution error is often more difficult to determine. If you cannot determine the cause of the error from the execution error message, you can use the interactive debug utility or interpret memory dumps to locate the cause. CYBER Interactive Debug is described in its reference manual (listed in the preface). This section describes central memory dumps and their use as a debugging aid.

CYBER 180, CYBER 170, CYBER 70 and 6000 Computer Systems that support NOS have the central exchange jump/monitor exchange jump (CEJ/MEJ) feature. This feature enables a program to directly switch CPU control to the system monitor. The information transferred from the CPU to central memory by an exchange jump operation is called an exchange package.

You can dump the job exchange package and locations within the job field length using the DMP, DMD, and DMB commands described in Volume 3, System Commands. Most CPU mode errors result in an exchange package dump.

You interpret a memory dump using the load map and the compiler-generated symbolic reference map. Dump interpretation may also require knowledge of display code equivalences (Volume 3, System Commands), machine codes, and internal integer and floating point number representations (refer to the COMPASS Reference Manual).

EXCHANGE PACKAGE DUMPS

You can dump a job's exchange package using a DMP, DMD, or DMB command within the job (refer to Volume 3, System Commands). Figures D-1 and D-2 show actual exchange package dumps. The format of the first dump is produced by all CYBER 170 Computer Systems except model 176; all CYBER 70 Computer Systems; and all 6000 Computer Systems. The second dump format is produced only by the model 176.

The following are the exchange package fields and their contents.

Label	Contents
P	Program address at which execution stopped.
RA	Reference address; starting address of central memory field length.
FL	Field length in central memory.

```
   EXCHANGE PACKAGE.

   P       242  A0  51760  B0      0      (A0)  0000 0000 0000 0000 0000
   RA   622400  A1      1  B1      1      (A1)  0516 0420 0000 0000 0000
   FL    52000  A2    114  B2     30      (A2)  0400 0005 0300 0007 7775
   EM     7007  A3    574  B3      6      (A3)  5555 5555 5555 5555 5555
   RAE       0  A4    557  B4     22      (A4)  7777 7777 7777 7777 7776
   FLE       0  A5    573  B5      1      (A5)  1717 0631 4631 4631 4632
   MA     3600  A6      1  B6   7776      (A6)  0516 0420 0000 0000 0000
                A7    277  B7  14657      (A7)  3232 3232 3206 0300 0171

   X0   7777 7777 7777 0000 0000
   X1   0000 0000 0000 0000 0000
   X2   0000 0000 0000 0000 0000
   X3   0000 0000 0040 0000 0000
   X4   2000 0000 0000 0000 0012
   X5   1717 0631 4631 4631 4632
   X6   0516 0420 0000 0000 0000
   X7   0000 0000 0000 0000 0000

(RA)    0000 0000 0000 0000 0000
(RA+1)  0516 0420 0000 0000 0000
```

Figure D-1. Exchange Package Dump

```
   EXCHANGE PACKAGE.

   P     10435  A0   2165  B0      0      (A0)  1725 2420 2524 0000 0131
   RA   136100  A1      1  B1      1      (A1)  0516 0420 0000 0000 0000
   FL    15000  A2   6251  B2 777755      (A2)  1717 0631 4631 4640 3615
   PSD   70000  A3      2  B3   6032      (A3)  0000 0000 0000 0000 0000
   RAE       0  A4   6207  B4  11437      (A4)  0400 0062 4600 0000 0000
   FLE       0  A5   4324  B5  12711      (A5)  2000 0000 0000 0000 0065
   MA     1200  A6      1  B6 776677      (A6)  0516 0420 0000 0000 0000
   EEA    1200  A7  12557  B7     30      (A7)  6000 0000 0000 0001 5000

   X0   0000 0000 0000 0000 0000
   X1   0000 0000 0000 0000 0000
   X2   1717 0631 4631 4640 3615
   X3   2000 0000 0000 0000 0012
   X4   2000 0000 0000 0000 0000
   X5   0000 0000 0000 0000 0000
   X6   0516 0420 0000 0000 0000
   X7   2000 0000 0000 0000 0001

(RA)    0000 0000 0000 0000 0000
(RA+1)  0516 0420 0000 0000 0000
```

Figure D-2. Exchange Package Dump for Model 176

Label	Contents
EM†	Exit mode. This field contains control bits and status bits. Each control bit set indicates that if the hardware-detected error occurs, the program aborts. A status bit(s) is set when the specified condition occurs, except for bit 4 on models 815, 825, 835, 845, and 855. On models 815, 825, 835, 845, and 855, bit 4 is the instruction stack purging status bit and indicates whether stack purging is extended (1) or normal (0). The bit positions are numbered with 0 as the rightmost bit (each digit shown represents 3 bits). For more information, refer to the MODE macro in Volume 4, Program Interface.

Bit Position	Error
11	CM data error.††
10	Central memory control (CMC) input error.
9	Extended memory flag register operation parity error.††
8	Central memory copy flag.
7	Reserved.
6	Software flag.
5	CMU interruption flag.
4	Instruction stack purge flag.†††
3-4	Hardware error exit status bits.††††
2	Indefinite operand.
1	Operand out of range.
0	Address out of range.

The EM field in figure D-1 has bit positions 11, 10, 9, 2, 1, and 0 set.

PSD††††† Program status designator (PSD) register. Each bit set indicates the setting of a mode flag or an error condition. The bit positions are numbered with 0 as the rightmost bit (each digit shown represents 3 bits).

Bit Position	Error
14	Indefinite mode.
13	Overflow mode.
12	Underflow mode.

†Does not apply to model 176.
††Applies to all CYBER 170 Computer Systems except models 176, 815, 825, 835, 845, and 855.
†††Applies to models 815, 825, 835, 845, and 855 only.
††††Applies to model 74 only.
†††††Applies to model 176 only.

Label	Contents	
	Bit Position	Error
	11	LCME error.
	10	CM error.
	9	LCME block range error.
	8	CM block range error.
	7	LCME direct range error.
	6	CM direct range error.
	5	Program range error.
	4	Not used.
	3	Step condition.
	2	Indefinite condition.
	1	Overflow condition.
	0	Underflow condition.

The PSD field in figure D-2 has bit positions 14, 13, and 12 set.

RAE	Extended memory reference address; starting address of extended memory field length.
FLE	Extended memory field length.
MA	Monitor address (normal exit address for the model 176).
EEA	Error exit address (model 176).†
Ai	Contents of the address registers.
(Ai)	Contents of the central memory word addressed by the named address register.
Bi	Contents of the increment registers.
Xi	Contents of the operand registers.
(RA)	Contents of the reference address word.
(RA+1)	Contents of the request word following the reference address word.

†Normally this does not apply to you if you are writing applications programs.

USING DUMPS

You receive an exchange package and partial CM dump when a hardware-detected error occurs. You can also obtain dumps of your job's exchange package and field length by including a DMP, DMB, or DMD command in your job. If you are using FORTRAN 5, you can generate a CM dump within a program using the DUMP or PDUMP subroutine (refer to the FORTRAN 5 Reference Manual). If you are using COMPASS, you can specify the REPRIEVE macro to control error processing and the SYSTEM macro to generate dumps (refer to Volume 4, Program Interface).

When the system hardware detects one of the error conditions listed in the MODE command description (refer to Volume 3, System Commands), NOS dumps the job exchange package and the contents of the 32 words preceding and the 32 words succeeding the address where the job step terminated.

You can specify commands after an EXIT command; the commands then are processed only if the job step terminates abnormally (refer to Volume 3, System Commands). If you specify a dump command to be executed and if the job step aborts, you can analyze the resulting dump to determine the cause of the job step abort.

Example:

This example shows the exchange package and central memory dump that results from a FORTRAN 5 execution error.

You submit the following job to compile and execute a FORTRAN 5 program called FPROG.

```
FJOB.
USER,username,password.
CHARGE,*.
GET,FPROG.
FTN5,I=FPROG.
MAP,PART.
LDSET,PRESET=ZERO.
LGO.
```

The FTN5 command compiles the source program in the retrieved file FPROG. The resulting program listing and symbolic reference map are shown in figure 14-3. The fields within the reference map are defined in the FORTRAN 5 Reference Manual.

```
1            PROGRAM FPROG      74/74   OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+564        yy/mm/dd. 14.10.13       PAGE     1
          DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,PL=5000
          FTN5,I=FPROG.


         1                PROGRAM F PROG
         2                DIMENSION N(10)
         3                DATA (N(I), I=1,10)/1,2,3,4,5,6,7,8,9,10/
         4                NSUM=0
         5                DO 10 I=1,111111
         6                NSUM=NSUM+N(I)
         7             10 CONTINUE
         8                STOP
         9                END


 -VARIABLE MAP--(LO=A)
  -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

   I          46B                              INTEGER
   N          34B                              INTEGER        10
   NSUM       47B                              INTEGER


 -STATEMENT LABELS--(LO=A)
  -LABEL-ADDRESS-----PROPERTIES----DEF

      10  INACTIVE   DO-TERM         7


 -ENTRY POINTS--(LO=A)
  -NAME---ADDRESS--ARGS---

   FPROG       5B     0


 -STATISTICS-

   PROGRAM-UNIT LENGTH              51B =  41
   CM STORAGE USED                61300B =  25280
   COMPILE TIME                   0.015 SECONDS
```

Figure D-3. Program Listing and Symbolic Reference Map

The MAP(PART) command instructs the loader to generate a partial load map when it loads the program. The LDSET(PRESET=ZERO) command tells the loader to set uninitialized memory words to zero during the next load. (Refer to the CYBER Loader Reference Manual for loader command descriptions.)

The LGO command loads and executes the object program that the compiler wrote on file LGO. The partial load map is shown in figure D-4. The fields within the load map are defined in the CYBER Loader Reference Manual.

Figure D-5 shows the job dayfile and figures D-6 and D-7 show the exchange package and central memory dump, respectively.

```
1          LOAD MAP - FPROG                                CYBER LOADER 1.5-564       yy/mm/dd. 14.10.14.       PAGE     1


          FWA OF THE LOAD            111
          LWA+1 OF THE LOAD         5746

          TRANSFER ADDRESS -- FPROG            116

          PROGRAM ENTRY POINTS --       FPROG            116


          PROGRAM AND BLOCK ASSIGNMENTS.

          BLOCK       ADDRESS   LENGTH    FILE     DATE     PROCSSR VER LEVEL  HARDWARE   COMMENTS

          FPROG           111       51    LGO      yy/mm/dd FTN     5.1 564    666X I        PROGRAMOPT=0,ROUND= A/ S/ M
          SYSAID=         162        1 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               LINK BETWEEN SYS=AID AND INITIALIZATION CODE.
          /FCL.C./        163       36
          /STP.END/       221        1
          /Q5.IO./        222      275
          Q5NTRY=         517       30 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               FCL5 - INITIALIZE FCL5 RUN TIME LIBRARY.
          CHMOVE=         547      162 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               CHARACTER MOVE AND CONCATENATE
          /FCL=ENT/       731       73
          FCL=FDL        1024       63 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               FCL CAPSULE LOADING
          FEIFST=        1107        3 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564                CONVERTED DATA STORAGE
          /AP.IO./       1112       17
          FORSYS=        1131     1371 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               FORTRAN OBJECT LIBRARY UTILITIES.
          FORUTL=        2522      200 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               FCL MISC. UTILITIES.
          GETFIT=        2722      173 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               GETFIT= - LOCATE A FIT GIVEN A UNIT DESCRIPTOR
          Q5RPV=         3115       14 SL-FTN5LIB  yy/mm/dd COMPASS 3.6 564               FCL5 - ABORT RECOVERY INITIALIZATION
          CPU.CPM        3131        5 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               yy/mm/dd. yy/mm/dd. CONTROL POINT MANAGER PROC
          CPU.LFM        3136       10 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               yy/mm/dd. yy/mm/dd. LOCAL FILE MANAGER PROCESS
          CPU.SYS        3146       40 SL-SYSLIB   yy/mm/dd COMPASS 3.6 552               PROCESS SYSTEM REQUEST.
          CMF.ALF        3206      162 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CMM V1.1 - ALLOCATE FIXED.
          CMF.CSF        3370        6 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CMM V1.1 - CHANGE SPECS FIXED.
          CMM.FFA        3376       14 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CMM V1.1 - FIXED FREE ALGORITHM.
          CMF.FRF        3412       36 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CMM V1.1 - FREE FIXED.
          CMF.GSS        3450       22 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CMM V1.1 - GET SUMMARY STATISTICS.
          CMM.MEM        3472        7 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564
          CMM.R          3501      207 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CMM V1.1 - RESIDENT SUBROUTINES.
          CMF.SLF        3710       22 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CMM V1.1 - SHRINK AT LWA FIXED.
          /FDL.COM/      3732       17
          FDL.RES        3751      212 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               FAST DYNAMIC LOADER RESIDENT.
          FDL.MMI        4163      250 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               FDL MEMORY MANAGER INTERFACE.
          CTL$RM         4433      473 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CRM CONTROLLING ROUTINE.
          CTL$SKP        5126       57 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CRM CONTROLLER - SKIP PHYSICAL/FILE.
          CTL$WR         5205       44 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CRM CONTROLLER - WEOX, REWIND
          ERR$RM         5251       25 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CRM ERROR PROCESSOR ENTRY.
          LIST$RM        5276       67 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CRM - ALLOCATE SPACE FOR LIST OF FILES
          RM$SYS=        5365        5 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               CRM - POST RA+1 REQUEST
          RECOVR         5372      354 SL-SYSLIB   yy/mm/dd COMPASS 3.6 564               RECOVR - V2.0, USER INTERFACE TO *RPV*.


               .070 CP SECONDS              23500B CM STORAGE USED                  4 TABLE MOVES
```

Figure D-4. Partial Load Map

```
14.10.09.FJOB.
14.10.09.USER,FREE007,.
14.10.09.CHARGE,*.
14.10.09.*  CHARGE(1223,AB3N456)
14.10.11.GET,FPROG.
14.10.13.FTN5,I=FPROG.
14.10.13.     61300 CM STORAGE USED.
14.10.13.     0.016 CP SECONDS COMPILATION TIME.
14.10.13.MAP,PART.
14.10.13.LDSET,PRESET=ZERO.
14.10.13.LGO.
14.10.15. OVER-INDEXED ARRAY IN FPROG   NEAR LINE 6
14.10.15.     10000 MAXIMUM EXECUTION FL.
14.10.15.     0.009 CP SECONDS EXECUTION TIME.
14.10.15. CPU ERROR EXIT AT  000130.
14.10.15. CM OUT OF RANGE.
14.10.16.UEAD,      0.002KUNS.
14.10.16.UEPF,      0.045KUNS.
14.10.16.UEMS,      1.840KUNS.
14.10.16.UECP,      0.109SECS.
14.10.16.AESR,      2.480UNTS.
14.10.16.$OUT(*/OP=E)
14.10.16. NO FILES PROCESSED.
14.10.16.$DAYFILE(OUTPUT,JT=D)
```

Figure D-5. Dayfile

```
   EXCHANGE PACKAGE.
   P          0  A0     113  B0       0   (A0) 0000 0000 0000 0100 0000
   RA    643300  A1     160  B1       1   (A1) 0353 6613 0571 3072 2764
   FL     10000  A2     157  B2       0   (A2) 0000 0000 0000 0000 5634
   EM      7007  A3    6000  B3       0   (A3) 5325 0437 0061 1000 0001
   RAE        0  A4    2560  B4    6001   (A4) 0000 0000 0000 0000 0000
   FLE        0  A5     177  B5     116   (A5) 0000 0000 0000 0001 1610
   MA      5200  A6     161  B6    6000   (A6) 0000 0000 0000 0032 3153
                 A7     157  B7       1   (A7) 0000 0000 0000 0000 5634

   X0   0000 0000 0000 0000 5634
   X1   0353 6613 0571 3072 2764
   X2   0000 0000 0000 0000 5634
   X3   0000 0000 0000 0000 0000
   X4   0000 0000 0000 0000 0000
   X5   0000 0000 0000 0001 1610
   X6   0353 6613 0571 3072 2764
   X7   0000 0000 0000 0000 5634

(RA)    0001 0001 3000 0000 0000
(RA+1)  0000 0000 0000 0000 0000
```

Figure D-6. Exchange Package Dump

```
CM DUMP  FROM      70 TO     170.
     70      14071 75700 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
     74      00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
    100      54000 00000 01000 00001    00563 50000 00000 05746    00000 00000 00000 00000    00000 00000 00000 00000
    104      00000 00000 00000 05746    00000 00000 00000 00000    00000 00000 00000 00000    00000 00000 00000 00000
    110      06202 21707 00000 00116    06202 21707 55550 00116    17770 00000 00000 00000    00000 00000 00010 00000
    114      00000 00000 00000 11610    61027 77726 61020 00111    51100 00113 01000 03115    61020 00004 13777 46000
    120      51700 00160 61000 46000    61020 00005 71600 00001    51600 00157 71100 00001    71203 31007 37021 10700
    124      51700 00161 61000 46000    61020 00006 51100 00160    51200 00157 52320 00144    36631 54610 61000 46000
    130      61020 00007 51100 00157    71200 00001 36021 10700    54710 51200 00161 46000    72127 77776 10611 54620
    134      03210 00125 61000 46000    61020 00010 51100 00141    01000 01441 00100 00111    61020 00011 61000 46000
    140      00000 00000 00000 00000    00000 00000 00000 00140    00000 00000 00000 00000    00000 00000 00000 00000
    144      00000 00000 00000 00000    00000 00000 00000 00001    00000 00000 00000 00002    00000 00000 00000 00003
    150      00000 00000 00000 00004    00000 00000 00000 00005    00000 00000 00000 00006    00000 00000 00000 00007
    154      00000 00000 00000 00010    00000 00000 00000 00011    00000 00000 00000 00012    00000 00000 00000 05634
    160      03536 61305 71307 22764    00000 00000 00003 23153    01000 02040 00000 00000    55555 55555 55555 55555
    164      40404 04040 40404 04040    11162 02524 00000 00000    17252 42025 24000 00000    17171 27432 14774 13155
    170      20001 20726 42717 30565    00000 63146 31463 14632    00000 00000 00000 00003    00000 00000 00000 00000
```

Figure D-7. Central Memory Dump

INDEX

Abort B-1
Absolute address B-1
Absolute Load B-1
Access point, file 4-8
Access subsystem 2-5
Acoustic coupler C-3,4,7; B-1
Alphanumeric characters B-1
AP command D-5
APPEND command 5-4; 7-7
Application B-1
ASCII B-1
ASCII character set A-1
ASCII code set A-1
ASCII command 4-8; A-3
ASCII graphic 128-character set A-3
ASCII graphic 63/64-character code set 3-8; A-1,2,3,4
ASCII graphic 95-character code set A-1,3,4
ASCII mode 4-8; A-3; B-1
ASCII 128-character set A-1,10
Asynchronous terminal C-7,14
ATTACH command 4-1,3; 5-1,8,14; 7-7
ATTN key C-10
AUTO command 2-17
Automatic recognition of terminal C-7
Auxiliary device B-1

Banner page 2-7
BASIC
  Definition B-1
  File sequence numbers 2-15
BASIC command 11-1,3
BASIC subsystem 2-5; 11-2
Batch job 2-1,3,5,7,12,13,14; 3-6,7; 4-8; 7-1
Batch job, deferred 2-5
Batch job, local 2-5
Batch job, remote 2-5
Batch subsystem 2-5,6
BCD, external A-12
BEGIN command 8-6; 9-25
Beginning-of-information (BOI) 3-3; 4-8; B-2
Binary information, copying 4-4
Bisynchronous terminal C-7,17
Bit B-2
BKSP command 4-8,11
Blank labeling, magnetic tape 6-1
Block B-2
Block mode transmission C-10
BOI (See Beginning-of-information)
Byte B-2

Card deck 2-5; 3-3
Carriage control 4-7
Carriage return key C-10
Catalog B-2
CATALOG command 10-4
CATLIST command 5-12,13,18,19,22
CDC graphic 63/64-character code set 3-8; A-1,2,3,4
CDC mode 4 terminal C-17
CEJ/MEJ (See Central exchange jump/monitor exchange jump)
Central exchange jump/monitor exchange jump (CEJ/MEJ) D-1
CHANGE command 5-17
Character B-2
Character sets A-1
  Batch A-3,7
  Interactive A-3,5
Character, uppercase/lowercase translation 4-8
CHARGE command 2-9,12,15
Charge number 2-9,21
Checklist patterns, procedure 9-11,15
Checklist, procedure 9-11,12
COBOL5 command 11-1
Code set
  6-bit display 4-8; A-1,2,3,4
  6/12-bit display 4-8; A-1,10
  7-bit ASCII 3-8; A-1,2,10
Code set conversion, magnetic tape A-12
Code sets, magnetic tape A-12
Code sets, print file 3-8
Coded file, copying 4-6,7,9
Command
  Definition B-2
  Format 2-11
  Parameters, order-dependent 2-11
  Parameters, order-independent 2-11
  Record B-2
  Separator 2-11
  Terminator 2-11,13
  X Prefix 2-6
Command, system 2-1
Commands, subsystem 2-5

COMPASS
    Error processing D-5
    REPRIEVE macro D-5
COMPASS command 11-1
Compilation, program 11-1
Compile B-2
Concatenation character (_) 9-6,7
Control key B-2
Control point B-2
Control registers, Flow control command
  8-10,11
Control statement B-2
COPY command 4-7; 6-4
COPYBF command 4-4
COPYBR command 4-4
COPYCF command 4-6
COPYCR command 4-6
COPYEI command 4-4
COPYSBF command 3-8; 4-7
CPU errors 9-10
CR key C-10
CTRL/T 2-17
CYBER Interactive Debug D-1
CYBER Loader Utility 1-1; 10-2;
  11-2,3,4,5,6; B-3
CYBER Record Manager (CRM) 3-2; B-3

Data set C-3,4,7
Dayfile B-3
DAYFILE command 7-5
Dayfile, job 7-4,5
Debugging aids D-1
Default B-3
Deferred batch job 2-5
DEFINE command 4-1; 5-1,8,12,14; 7-7
Device B-3
Dial-up terminal C-7
Direct access permanent file 3-9; 4-3;
  5-1,8,9,14; B-3
Directive, definition B-3
Display code A-1; B-3
DISPLAY command 8-8,11,13
DMB command D-5
DMD command D-5
DMP command D-5
Dump, memory D-1,5
DUMP subroutine, FORTRAN D-5
Duplex transmission B-3

EBCDIC code B-3
ELSE command 8-3
Empty PRU/record B-3
End-of-file (EOF) 3-4; B-3
    Submit file 2-15
End-of-information (EOI) 3-4; 4-4; B-4
End-of-record (EOR) 3-4; B-4
    Submit file 2-15
End-of-tape (EOT) B-4
.ENDHELP directive, Flow control command
  9-19
ENDIF command 8-1,3
.ENDIF directive, Flow control command 9-20
ENDW command 8-5
ENQUIRE command 2-19,23
Entry point B-4
EOF (See End-of-file)
EOI (See End-of-information)
EOR (See End-of-record)
EOT (See End-of-tape)
EOT key C-10
Error flag, definition B-4
Error processing 7-1,7; D-5
    Hardware errors 7-8
    Memory dump D-1
    No-abort option 7-7
    Wait-if-busy option 7-7
Error processing in procedures 9-8
ETX key C-10
Exchange package 7-8; B-4; D-1,5
Executable object code B-4
Execute-only file B-4
Execute subsystem 2-5
Execution, program 11-1
EXIT command 7-1; 9-9
Expressions, Flow control command
  8-8,9,10; 9-7
External reference B-4

Family Device B-4
Family name 2-8,10; B-4
FCOPY command 3-8; A-3,12
Field length B-4
File B-5
    Alternate user access 5-12,13,16,17
    Block size 5-1
    Catalogue information 5-18,19
    Coded 4-6,7,9
    Copying 4-4,5
    Definition 2-2
    Execution 11-1
    Positioning 4-8
    Record extraction 10-7
    Releasing 4-13,14,15
    Rewinding 4-12
    Routing 3-7
    Semiprivate 5-12,16,22
File access mode 5-12,13,14,16,17
File access point 4-8
File attributes 8-12
File, direct access 3-9; 4-3; 5-1,8,9,14

FILE function 8-12,14
File header label (HDR1) 6-3,14
File identifier, magnetic tape 6-16
File, indirect access 3-9; 4-2;
5-1,2,3,4,6,13
File INPUT 3-5,6
File, input 2-12
File, library 10-1
File, local 3-9; 4-1,2,13,15
File, multifile 2-15; 3-1,2; 4-4,6,10
File, multirecord 10-7
File, name 3-2; 4-13; 5-17
File OUTPUT 3-5,6
File, owner 5-2
File password 5-13,17
File, permanent 3-9; 5-1
File permit category 5-12,17
File, primary 4-1,2
File, print 3-8
File, private 5-12,16,22
File, public 5-12
File PUNCH 3-7
File, punch 3-7
File PUNCHB 3-7
File P8 3-7
File, set B-5
File, structure 3-1,2,3
File, temporary 3-9
Flag B-5
Flow control command
Control registers 8-1,10
Definition 8-1
.ENDHELP directive 9-19
.ENDIF directive 9-20
Expressions 8-8,9,10
Functions 8-12
.HELP directive 9-19
.IF directive 9-20
Operators 8-8
Summary table 8-3
Symbolic names 8-8,9
FORTRAN
Definition B-5
Memory dump D-5
FORTRAN command 11-3
FORTRAN DUMP subroutine D-5
FORTRAN PDUMP subroutine D-5
FORTRAN subsystem 2-5; 11-2
FSE (See Full Screen Editor)
FTN command 11-1
FTNTS subsystem 2-6
FTN5 command 11-1
Full Screen Editor (FSE) 1-2; 2-2; 4-1;
12-1,2; B-5
Copying lines 12-10
Creating files 12-3
Directives 12-5
Inserting/deleting text 12-8
Line mode 13-1
Locating text strings 12-8
Online help 12-12
Screen mode 13-1
UNDO directive 12-9
Function keys, programmable 12-1
Functions, Flow control command 8-12

GE tape density 6-2
GET command 4-1,2; 5-1,13; 7-7
Global library set 10-2; 11-9,10,11; B-5
GTR command 10-7

Hardwired terminal B-5; C-7
HASP terminal C-15
HD command C-20
HD tape density 6-2
HDR1 label (See Magnetic tape)
.HELP directive, Flow control command 9-19
HI tape density 6-2
HN command C-19
Host availability display (HAD) C-20
Host computer 1-2; 2-10; B-5; C-1,5
Host computer selection C-18,19,20
Host Path B-5
HS command C-18
HY tape density 6-2

IF command 8-1,3,8
.IF directive, Flow control command 9-20
Indirect access permanent file 3-9; 4-2;
5-1,2,3,4,6,13; B-6
Inhibit character (#) 9-5
Input file structure 2-12
Interactive Facility (IAF) B-6; C-28
Interactive job 2-1,4,5,10,13; 3-5,6
Interactive procedure 9-10,11,20
Interruption sequence B-6

Job
Batch 2-1,3,5,7,12,13,14; 3-6,7; 4-8;
7-1
Batch, local 2-5
Batch, remote 2-5
Default destination 2-20
Deferred batch 2-5
Definition 2-1; B-6
Interactive 2-1,4,5,10,13; 3-5,6
Status 2-19,20
Structure 2-7

Submission 2-14; 3-7
Termination 2-13
Job command 2-7,12
Job connection status 2-20
Job dayfile 7-4,5
Job name 2-7
Job sequence name (JSN) 2-10,13,20; B-6
Job service class 2-20
JSN (See Job sequence name)
Job step B-6

Label B-6
LABEL command 4-1; 6-2,6
Labeled tape B-6
LDSET command 11-9,10
LIBEDIT command 10-1
Directives 10-8
Format 10-9
LIBGEN command 10-1,3,4,6
Libraries 1-1
Library
Catalogue of contents 10-4,5
Definition 10-1
Global library set 10-2; 11-9,10,11
Loading programs from 11-9
Local library set 10-2; 11-9,10,11
Program 10-1; 13-5,12,16
Search order 11-11,12
Source code 10-1
System 10-2; 11-9
User name 10-2
User (See User Library) 10-1
LIBRARY command 11-9
LID (See Mainframe logical identifier) 2-20
LIMITS command 2-21
LINE command 12-1
Line Mode B-6
Line printer A-3,4
LISTLB command 6-16
LIST80 command 3-7
Literal delimiter 9-7,8
Literal string 9-7,8
LO tape density 6-2
LOAD command 11-6
Load map B-6; D-1
Loading, interactive program 11-9
Loading, program 11-2,6,7
Loading sequence B-6
Local batch job 2-5
Local file 3-9; 4-1,2,15; B-7
Local library set 10-2; 11-9,10,11; B-7
Locked File B-7
Logical device B-7
Logical record B-7
Login 2-10; B-7; C-1,24,25,26,30
Login display 2-10
Login sequence 2-10
Logout 2-13; B-7
Logout command 2-13
Logout display 2-13
LO72 command 3-7

Magnetic tape
ANSI-labeled 6-3
Blank labeling 6-1
Code set conversion A-12,13,14,15
Copying 6-4
File header label (HDR1) 6-3,14
File identifier 6-16
Files 6-1
Format 6-3
Multifile set 6-14
Multivolume files 6-12
Reading 6-2
Reading labels 6-16
Sequence number 6-14,16
Set identifier 6-14,16
Supported code sets A-12
Tape density 6-2
Volume header label (VOL1) 6-1
Volume serial number (VSN) 6-1
Writing 6-6,12
Mainframe logical identifier (LID) 2-20
Mass storage B-7
Master device B-7
Menu procedures 9-11,20; B-7
Menu procedures, procedure header directive 9-22
Message Control System (MCS) C-28
Modem C-3,4,7
Modify Utility 1-2; 10-1; 13-4,5
Calling 13-12
Compile file 13-14
Deck 13-10
Directives 13-6
File structure 13-6
Output 13-6
Multifile file 2-15; 3-1,2; 4-4,6,10; B-7
Multifile set 6-14; B-7
Multihost arrangement B-7
Multirecord file 9-22

Name call command 11-6; B-7
Name call loading 11-6
Network B-7
Application selection C-27,31
Applications C-28,39
Hardware configurations 1-2
Network Control Character B-7
Network Operating System (NOS) 1-2; B-8

NEW command 4-1,2,13
NEWLINE key C-10
NEXT key C-10
No-abort option 7-7
No-auto-drop status 4-2,13
Node, host 1-2
NOEXIT command 7-2
Nonallocatable device B-8
Noninteractive procedures 9-10
NORMAL command 4-8; A-3
Normal mode 4-8; A-3; B-8
NOS Screen Formatting 12-1
NOS (See Network Operating System) 1-2
Null subsystem 2-6
NUMBER function 8-12

Object code B-8
Object module B-8
OLD command 4-1,2,13; 5-1; 7-7
ONEXIT command 7-3
Online help
    Full Screen Editor (FSE) 12-12
    Interactive procedures 9-10,14,19
    Menu procedures 12-26
    Procedures in screen mode 12-20
Order-dependent parameters 2-11; B-8
Order-independent parameters 2-11; B-8
Origin Type, job B-8
Output
    Disposition 2-14; 3-6,7
    Listing 7-6
    Print file formatting 4-7
OVCAP B-8
Overlay B-8

Packet switching network (PSN) C-7
Parameter B-8
    Order-dependent B-8
    Order-independent B-8
Parity B-8
Password 2-1,8,10; B-8
PDUMP subroutine, FORTRAN D-5
PE tape density 6-2
Permanent file family B-9
Permanent file, definition 3-9; B-8
Permanent file device B-8
PERMIT command 5-12,16
Physical record unit (PRU) 3-3; 5-1; B-9
PLATO C-28
Prefix character B-9
Primary file 4-1,2; B-9
Print file formatting 4-7
Printer, central site 3-8
Printer, line A-3,4

.PROC directive 9-2,3,12
Procedure
    Checklist 9-12
    Checklist patterns 9-11,15
    Command terminator 2-13
    Commands 8-6
    Concatenation character (_) 9-6,7
    Definition 9-1; B-9
    Error processing 7-1; 9-8,9
    Execution 9-2,25
    Expression 9-7
    File B-9
    Inhibit character (#) 9-5,6
    Interactive 9-1,10,11,20
    Menu 9-11,20,22; 12-26
    Multirecord file 9-22,25,26
    Nested 9-8
    Noninteractive 9-10
    Online help 9-10,12,14,19,23; 12-20,26
    Parameter default values 9-16
    Parameter substitution
      9-3,4,5,6,7,11,12,15
    Screen mode display 12-23
    Structure 9-13
    Termination 9-8,10
Procedure header directive, interactive
  format 9-2,3,12
Procedure header directive, menu format
  9-22
Program
    Compilation 11-1
    Debugging D-1
    Execution 11-1
    Loading 11-2,6,7
    Loading from library 11-9
    Loading, interactive 11-9
Program library 10-1; 13-5,12,16
Project number 2-9,21
Prologue procedures 9-22,27; B-9
Protocol, line C-7
PRU (See Physical record unit)
PSN (See Packet switching network)
Punch files 3-7
PURGE command 5-10; 7-7

QGET command 2-14

Random access file B-9
Read-only permission B-9
Record 3-3
    Copying 4-4
    Definition B-9
Reference address B-9
Relative address B-9

Remote Batch Facility (RBF) C-28
Remote batch job 2-5
Remote batch terminal 2-14; C-9
RENAME command 4-13
REPLACE command 5-1,2,6
REPRIEVE macro, COMPASS D-5
RESOURC command 6-9
RET key C-10
RETURN command 4-13,14; 6-10
RETURN key C-10
REVERT,ABORT command 9-10
REVERT command 8-6; 9-8,9
REWIND command 4-8,12
ROUTE command 3-7; 7-6

Satisfying external references B-9
SAVE command 5-1,2,12,13
SCOPY command 4-7
SCREEN command 12-1
Screen formatting 12-1
Screen mode 1-2; B-10
  Procedure display 12-14,23
  Terminal support 12-1
Secured System B-10
Segment B-10
Segmentation B-10
SEND key C-10
Sequence number B-10
Sequence number, magnetic tape 6-14,16
Sequential, file B-10
SET command 8-8,11,13
Set identifier, magnetic tape 6-14,16
SETFS command 4-13
SETJSL command 2-23
SETTL command 2-23
Single-Host Arrangement B-10
SKIP command 8-1
SKIPEI command 4-8,11
SKIPF command 4-8,10
SKIPFB command 4-8,12
SKIPR command 4-8,10
SLOAD command 11-7
Source code B-10
Source code library 10-1
Source code maintenance utilities 13-1,4,5
SRU limit 2-23
Standard labeled tape B-10
String, character B-10
  Literal 8-12,14; 9-7,8
  Numeric 8-12,14
SUBMIT command 2-14,15; 7-6
  Directives 2-14
Submit file 2-13,15; B-10
  Character translation 4-8
  Creation 2-17
  Line sequence numbers 2-15,17
Subsystem 2-5
Symbolic names, Flow control command 8-9
SYSLIB 10-2; 11-9,11
System
  Command 2-1
  Definition B-10
  Features 1-1
  File B-11
  Resources 2-1,9,13
  Secured 1-4
System library 10-2; 11-9; B-11
System logical record 3-1,3
System resource unit 2-23; B-11
System resources
  Limits 2-21,23
  Time limit 2-23

Tape (See Magnetic tape)
Tape Verification Facility (TVF) C-28
Temporary file 3-9; B-11
Terminal
  Asynchronous C-7,14
  Automatic recognition of C-7
  Bisynchronous C-7,17
  CDC mode 4 C-17
  Connection to network C-3,13
  Dial-up C-7
  Hardwired C-7
  HASP C-15
  Name B-11
  Protocol C-7
  Screen mode support 12-1
  Session B-11
  Set-up C-2,12
Terminal Classes C-8
Terminal, remote batch 2-14
Termination sequence 2-17; B-11
Transaction Facility (TAF) C-28

UJN (See User job name)
UNLOAD command 4-13,15; 6-10
Update Utility 13-4
User index B-11
User index hash B-12
USER command 2-7,8,12,15
User job name (UJN) 2-7; B-12
User library 10-1
  Catalogue of contents 10-4
  Creation 10-3,4
  Editing 10-7
  Updating 10-6,11
User name 2-1,10; 5-2; B-12
User name library 10-2

Validation file B-12
Validation, user 2-1
Volume header label (VOL1) 6-1
Volume serial number (VSN) 6-1
VSN command 6-12
VSN (See Volume serial number)

Wait-if-busy option 7-7
WHILE command 8-5,8
Write interlock B-12
Write mode 5-9
Write ring 6-3; B-12

XMIT key C-10
X prefix, system command 2-6

ZZZDUMP 7-8

029 punch code A-2
6-bit display code A-1,2,3,4,12
6/12-bit display code A-1,3,4,10,12
6/7/8/9 card 2-12,13,15
7-bit ASCII code set 3-8; A-1,2,3,4,10,12
7/8/9 card 2-12,15

COMMENT SHEET

MANUAL TITLE: CDC NOS Version 2 Reference Set, Volume 2, Guide to System Usage

PUBLICATION NO.: 60459670 **REVISION:** C

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply ☐ No Reply Necessary

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

OLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

LD FOLD

CORPORATE HEADQUARTERS, P.O. BOX O, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION